DTIC FILE COPY

Technical Report 727

AD-A180 292

An Application of Simulated Annealing to Scheduling Army Unit Training

Roland J. Hart and Dwight J. Goehring

ARI Field Unit at Presidio of Monterey, California
Training Research Laboratory

U. S. Army

Research Institute for the Behavioral and Social Sciences

October 1986

87 5 13 059

U. S. ARMY RESEARCH INSTITUTE

FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the

Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

WM. DARRYL HENDERSON
COL, IN
Commanding

Technical review by

Katherine J. Griffin, Army Development and Employment Agency

W. Bruce Olson, U.S. Army Training Board

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A180292

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARI Technical Report 727	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN APPLICATION OF SIMULATED ANNEALING TO SCHEDULING ARMY UNIT TRAINING		5. TYPE OF REPORT & PERIOD COVERED Final Report February 1985-November 1985 6. PERFORMING ORG. REPORT NUMBER --
7. AUTHOR(s) Roland J. Hart and Dwight J. Goehring		8. CONTRACT OR GRANT NUMBER(s) --
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Research Institute Field Unit P.O. Box 5787 Presidio of Monterey, CA 93944-5011		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2Q263743A794 4413 100
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Institute for the Behavioral and Social Sciences 5001 Eisenhower Avenue, Alexandria, VA 22333-5600		12. REPORT DATE October 1986 13. NUMBER OF PAGES 106
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) --		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE --

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

--

18. SUPPLEMENTARY NOTES

--

Keywords:

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Training scheduling,
Unit training management,
Automated scheduling,
Automated training management,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Automated scheduling of Army training can reduce the complexity of scheduling the many required training and nontraining activities for a unit and lead to better planning of training. The objective of the research described here was to develop a prototype program for applying a particular heuristic called "simulated annealing" to the scheduling of Army unit training. The report presents an analysis of scheduling needs in Army units and describes the simulated annealing approach to Army scheduling. The
(continued)

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ARI Technical Report 727

20. (Continued)

characteristics and operations of the implementing algorithm are detailed with the computer code as an Appendix. The approach was judged as promising and warranting further research. Several augmentations of the current implementation of the simulated annealing approach are suggested.

FLD 19

Technical Report 727

An Application of Simulated Annealing to Scheduling Army Unit Training

Roland J. Hart and Dwight J. Goehring

ARI Field Unit at Presidio of Monterey, California
Richard K. Williams, Acting Chief

Training Research Laboratory
Jack H. Hiller, Director

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES
5001 Eisenhower Avenue, Alexandria, Virginia 22333-5600

Office, Deputy Chief of Staff for Personnel
Department of the Army

October 1986

Army Project Number
2Q263743A794

Education and Training

ARI Research Reports and Technical Reports are intended for sponsors of R&D tasks and for other research and military agencies. Any findings ready for implementation at the time of publication are presented in the last part of the Brief. Upon completion of a major phase of the task, formal recommendations for official action normally are conveyed to appropriate military agencies by briefing or Disposition Form.

FOREWORD

The Army Research Institute (ARI), Presidio of Monterey Field Unit, has conducted research on the use of computer technology for automating Army unit training management. This research included examining approaches for providing computer assistance to unit leaders for the scheduling of training. In the current effort, a prototype computer program was developed that employs a new heuristic approach called simulated annealing for solving the complex problems entailed in scheduling Army unit training. Research in this area was conducted under the sponsorship of the U.S. Army Training Board (ATB). This research is intended to lay the groundwork for an applications version of training scheduling software for the Integrated Training Management System (ITMS) currently under prototype development by the Army Development and Employment Agency (ADEA) at Fort Lewis, Washington.

EDGAR M. JOHNSON
Technical Director

AN APPLICATION OF SIMULATED ANNEALING TO SCHEDULING ARMY UNIT TRAINING

EXECUTIVE SUMMARY

Requirement:

In Army units, numerous required training and nontraining activities compete for time. The scheduling of training in the Army is complex when done manually, but computer assistance can reduce the complexity of training scheduling for planners and lead to better planning of training. Improved planning can make more efficient use of time and resources, and thus contribute to improving readiness.

Our objective was to develop a prototype program applying a particular heuristic called simulated annealing to the Army unit training scheduling problem.

Procedure:

The Army training scheduling problem is complex, in part because it occurs across different echelons and involves three different types of calendars/schedules. Priorities of higher echelons almost always predominate over those of lower echelons. In addition, many constraints exist for the Army training scheduling problem. Training and nontraining tasks require time and resources. Resources can be either reusable (e.g., ranges) or expandable (e.g., ammunition). Further, activities may have temporal constraints, such as requiring some tasks to be trained ahead of others, or co-occurrence constraints requiring some tasks to be trained simultaneously with others. There are also command priority constraints at each echelon that affect scheduling requirements. In addition, the priority given training varies at different times. To provide adequate computer-assisted training scheduling, the full diversity of the Army's scheduling problem must be accommodated.

Simulated annealing was selected as a method with high potential for dealing with both the complexity and heterogeneity of the problem. Simulated annealing is appropriate when, as with scheduling, multiple "good" solutions are acceptable as opposed to an uniquely "best" or optimal solution. The implementing algorithm operates iteratively by (1) random assignment of activities to times, (2) extensive modification due to constraint accommodations, and (3) evaluation of the schedule set using a cost function. The constraint accommodation proceeds by satisfying the most important constraints first, followed by the less important constraints. The cost value is used to determine the extent to which improvement in schedule set quality has been achieved.

Findings:

A scheduling system was written as a test program designed to permit evaluation of the feasibility of simulated annealing for solving the Army

training scheduling problem. The system was successful in demonstratng that this approach is a promising one that warrants further research. Several augmentations of the current implementation of the simulated annealing approach are suggested.

Utilization of Findings:

It appears highly likely that an applications version of a simulated annealing system can be developed and implemented on the Integrated Training Management System (ITMS), an automated unit management system at Fort Lewis, Washington. User friendliness and generation of adaptable reports for an applications scheduling system will be facilitated by the use of a relational data base management system (RDBMS).

AN APPLICATION OF SIMULATED ANNEALING TO SCHEDULING ARMY UNIT TRAINING

CONTENTS

LIST OF TABLES

LIST OF FIGURES

AN APPLICATION OF SIMULATED ANNEALING TO SCHEDULING ARMY UNIT TRAINING

This report describes research that is part of an ongoing project ultimately intended to provide computer assistance to Army units for the scheduling of training. The objective here is to develop a prototype program applying a particular heuristic called simulated annealing to the Army unit training scheduling problem.

The Army scheduling problem is characterized by competition between training and non-training tasks for scarce time and material resources. The large number of training and non-training tasks and their associated time and resource requirements make scheduling extremely difficult. The scheduling problem is further complicated by the following facts: (a) different echelons are involved in creating the schedules and calendars, (b) different calendars are used to plan for different lengths of time into the future, (c) a variety of scarce resources are required for training, and different echelons may control different resources, and finally (d) a wide variety of different types of scheduling constraints are involved, including requirements for the inclusion of command priorities and the inclusion of some activities as prerequisite for others.

Computer automated assistance for scheduling training offers a potential for improving the allocation of training time and resources in addition to saving time of planners responsible for accomplishing the scheduling task. Better allocation of training time to training tasks and more efficient use of scarce training resources are expected to yield increased unit readiness.

An additional advantage of computer automated assistance is its potential to help planners with comparison planning. Comparison planning involves the ability to ask "what if" questions, and then compare the resultant schedules based on alternate options. For example, higher echelons may not fully understand the impact on lower echelon schedules of schedule changes that are made at higher levels. However, computer automated assistance would provide the opportunity for higher echelons to understand the relative costs of making possible changes prior to actually making them. Thus, the impact of schedule changes can be identified and negative effects minimized.

The scheduling system described here is designed to lead to a system to be implemented on the Integrated Training Management System (ITMS), at Fort Lewis, Washington (see Army Development and Employment Agency [ADEA], January, 1985). The ITMS is a division-level prototype system demonstrating computer automation of unit training management. A system like the prototype may eventually be implemented widely throughout the Army. The Army training scheduling problem exists in its full complexity on the ITMS.

The scheduling system that is described here builds upon previous work on automated training scheduling and resource allocation, sponsored by the Army Training Board (Van der Eijk, Ignizio, & Yang, 1981; Yang & Ignizio, 1982; Medeiros & Yang, 1983; Medeiros & Yang, 1983b). This work includes a computer program designed to schedule training during mobilization called "The

Mobilization Resources Scheduling Program." This previous work was not designed specifically for implementation on ITMS and for this reason does not address the Army training scheduling problem in its full complexity as a multi-schedule, multi-echelon problem. Further, the previous efforts address many but not all of the training scheduling constraints important in a practical system.

The following discussion is organized into three sections. First, the Army training scheduling problem is described. Second, the use of simulated annealing as a solution for the scheduling problem is illustrated. The third section presents conclusions and recommendations emerging from the application of this approach. The computer program listing and a sample program run with data tables can be found as appendices to the report.

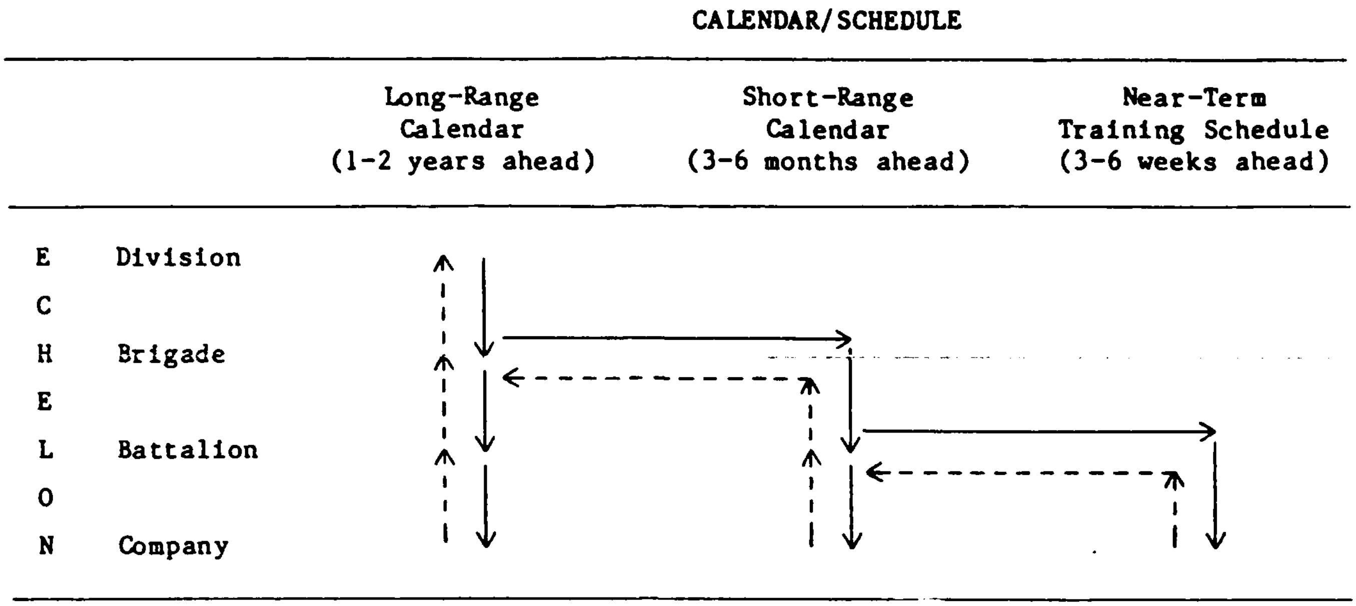
DESCRIPTION OF ARMY UNIT TRAINING SCHEDULING PROBLEM

Types of Schedules for Different Echelons

The training scheduling problem is defined here for all echelons within an Army division (see ADEA, 1985). The echelons within an Army division can be ordered from smallest to largest: company (about 150 soldiers); battalion (about 750 soldiers); brigade (about 3,000 soldiers); division (about 15,000 soldiers). There are three different training calendars/schedules that differ primarily in how far into the future needs are projected. These training calendars/schedules have been denoted "Long-Range Calendar" with projections up to 24 months in the future, "Short-Range Calendar" with projections from three to six months in the future, and "Near-Term Training Schedules" that extend from three to six weeks in the future. Different echelons have primary responsibility for preparing and updating the three types of calendars/schedules but the focus is at the battalion level. The relationships between echelons and the three types of calendars/schedules are shown in Figure 1.

The strict hierarchical organizational structure of the Army gives a top-down character to the process of training scheduling. Higher echelons may prescribe specific training or other activities. Major training events that originate at the division or higher level are first placed on the Long-Range Calendar. The predominant time units employed in the Long-Range Calendar are weeks, but specifications of days, such as holidays, also appear. Division has primary responsibility for preparation and revision of the Long-Range Calendar, although subordinate brigades and battalions may also be consulted. Typically the interest of the brigades and battalions is limited to those parts of the overall long-range calendar which concern them. Division has the responsibility for overall consistency and resolution of conflicts between subordinate units.

The Short-Range Calendar is prepared primarily at the battalion with input from the brigade. The predominant unit of time employed is days. The Short-Range Calendar takes mandated activities and adds them to the Short-Range Calendar, but with increased specificity and detail. Short-Range Calendars are coordinated with division. However, Short-Range Calendars are ultimately subordinate to decisions and event scheduling occurring at higher echelons.



- - - -> = recommendation
———> = decision
horizontal lines indicate transfers (or recommended transfers) of events between calendars/schedules

Figure 1. The Relationship Between Echelons and Schedules for ITMS System

Planning for the allocation of training resources like ranges, ammunition, MILES equipment, and training devices is accomplished on Long- and Short-Range Calendars.

Near-term Training Schedules are prepared and maintained at the company level. Training schedules depict, on a weekly basis, specific training activities in great detail. The predominant unit of time employed goes down to hours and even minutes. Many activities on Training Schedules are derived from Short-Range Calendars. Training Schedules are posted for the purpose of providing information to the members of the company, so that they can perform the activities. For this reason Training Schedules contain specific information not found on Short-Range Calendars. The detailed information typically includes precise times, locations, trainers and trainees involved, type of uniform, references, and remarks. The Training Schedule for each company is coordinated at the level of the parent battalion. Possible inconsistencies within a schedule are identified and conflicts between company-level Training Schedules are resolved. A typical conflict might involve two companies that request the same battalion classroom for the same time period.

The overall scheduling task is typified by the downward inheritance of events and activities prescribed by higher organizational echelons, coupled with additions, augmentation, and greater specificity at each successively lower level. There is an iterative process of allocating time to activities. Conflicts in allocation are resolved at the echelon just above the conflict. Activities scheduled vary in terms of command priority, resource requirements, relationships to other activities, and size. The size of scheduled activities may vary from a brigade level field training exercise to individual level skill training on specific tasks like first aid.

One practical aspect of the Army training scheduling problem that is important to appreciate is the necessity for frequent rescheduling. For a variety of reasons, activities dictated by higher echelons often preempt activities scheduled on company-level Training Schedules or unanticipated circumstances disrupt Training Schedules, even after the company level activities have been approved and considered final. Thus, at the company level rescheduling is common and requires decisions regarding which company-level training activities to reschedule as soon as possible, which ones to postpone, and which ones to eliminate. Assistance with rescheduling company-level training activities is a necessary practical feature for an automated scheduling system. When carrying out the rescheduling of activities, an automated system should minimize, as much as possible changes to existing calendars and schedules.

Types of Scheduling Constraints

There are distinct classes of scheduling constraints that shape Army training calendars and schedules. These contraints influence scheduling decisions differently and vary in their relative priority for being met. Satisfaction of relevant constraints serves as a useful criterion for judging the quality of training calendars. That is, relatively "good" schedules violate few scheduling constraints, while relatively "poor" schedules violate many contraints. Thus, constraint violations can be used in an automated

system to construct a "cost function"[1] for alternative calendars generated by the computer. These constraints, described below, relate to: time availability, downward inheritance, command priorities, inter-schedule compatibility and intra-schedule compatibility.

Time availability. The most basic factor affecting scheduling is time available for training. A sufficient amount of time to accommodate all of the training that ideally needs to be performed simply does not exist. This means that some desirable training activities must, of necessity, be left out of the Training Schedule thereby reducing the quality or increasing the cost of a schedule. Further, training activities vary in their difficulty for scheduling based upon the length of time required for completion. Longer activities are more difficult to schedule whereas activities of short duration can be fit in more easily. Thus, other factors remaining constant, those activities using more time can be considered "more expensive" if they cannot be fit into the schedule. Therefore, if it becomes impossible to include these longer tasks in the Training Schedule, their omission will add more to the cost function than the omission of shorter duration activities.

Downward inheritance. Activities or events are often fixed at specified times in the calendars and schedules by higher echelons and inherited downward. Thus, lower echelons inherit tasks or events specified by higher echelons. Such activities may be assigned a specific time or not. If times are specified, then lower level calendars and schedules must maintain the times established by higher echelons. Otherwise the lower echelons can assign the activities to times. Activity times may also be mandated in terms of fixed blocks of time. Subsequent scheduling of activities at a given echelon must be accomplished without alteration of activities that have been inherited from superordinate echelons. In addition, activities can be fixed at the current echelon. This means that all activities must be scheduled without altering these activities fixed at the current echelon as well as additional activities that may have been inherited from above. When conflicts arise that cannot be resolved, and activities remain unscheduled, cost is incurred.

Command priorities. The training priorities of commanders or their designated representatives at each organizational level differ and must be taken into account in the development of calendars and schedules. In the case of conflict in priorities for special training activities, the command priorities of higher echelons predominate over priorities of commanders at subordinate echelons, although communication between echelons may be possible to resolve differences. Under some conditions, particular units may be given higher priorities than other units for all training activities and resources. This situation might occur, for example, when some units must be prepared to mobilize more quickly than others.

---

[1]A cost function produces numbers that can be used to judge the relative quality of schedules. These numbers should not be taken literally as budget costs in dollars and cents for accomplishing training.

Inter-schedule compatibility. Since company-level units each have their own Training Schedules, conflicts can arise between Training Schedules for different units. The most frequent source of conflicts arises from requirements for scarce training resources. Resource requirements fall into two categories: expendable resources such as ammunition and gasoline, and renewable resources such as training facilities and equipment. Expendable resources are diminished in amount as a result of use, while renewable resources are not. Conflicts for expendable resources arise from shortages in supply. Eliminating expendable resource conflict requires increasing supplies or decreasing demands. Conflicts for renewable resources arise when demand at a particular time exceeds availability. Thus, renewable resources may be scarce at one time but not another depending upon the timing of demands. In this case, it is possible to eliminate the conflict by reassigning the times given to competing activities in different Training Schedules.

Intra-schedule compatibility. Conflicts can arise in the scheduling of events within a unit's Training Schedule. These conflicts may arise when there are: 1) temporal relationships between events, 2) temporal requirements for specific events, or 3) restrictions on the use of particular time blocks for training. A frequent temporal relationship between events occurs when training activities serve as prerequisites for other activities and must be trained ahead of the other activities yielding a precedence relationship. Violating such temporal relationships between events creates conflict within a Training Schedule. In other cases, specific training activities must occur together, producing synchronous or co-occurrence constraints for scheduled activities. Other temporal relationships between events are also possible, such as activities that must occur contiguously. Temporal requirements for specific events occur when activities have particular time-related requirements, such as darkness for training of night vision devices. The above temporal constraints may be established by commanders or pre-established in a data base. The data base may use doctrine, policy, and logical necessity as criteria for temporal constraint relationships.

Restrictions on time blocks for training generally occur at division level. Time is often blocked into groups of weeks labeled red, green, and yellow time that indicate the priority of training activities for subordinate units during a given block of time. In green time, training activities and events have priority for a unit, while in red time garrison support activities have priority. During yellow time (when this block exists) training activities have an intermediate priority. Intra-schedule conflicts are created when restrictions on the use of blocks of time are violated.

Input/Output Requirements for Scheduling

An automated system for scheduling can be divided into two distinct parts: an algorithmic and a human interface component. The algorithmic component employs an optimization methodology designed to minimize the cost function. Thus, the algorithmic part must be based on an appropriate problem definition and cost function. The human interface component involves program input and output.

While the input and output requirements are vital for any operational implementation of an automated scheduling system, the algorithmic aspects are a natural prerequisite. Thus, the primary focus of this paper is on the algorithmic aspect of the problem, including definition of the problem and description of the algorithm. The general feasibility of the approach is tested using an initial implementation of algorithm concepts. However, a brief discussion of requirements for input and output is presented first in order to provide an operational view of the applications system.

The primary requirement for input to and output of a scheduling system is that it be "user friendly". In order to create a practical system that will be used within Army units, it is important to keep program commands and data entry as simple and easy to learn as possible. In addition, the scheduling system should generate output information including summary reports, tailored to specific users' needs. In order to facilitate user friendliness and generate useful summary reports, the input/output component of the scheduling system will employ a relational data base management system (DBMS). A relational DBMS system is designed to minimize input-output requirements for program users. A possible criterion of scheduler user friendliness at the unit level is that no dedicated personnel are required.

Input to the automated scheduling system will be facilitated by a menu-driven database sub-system containing tables or lists of training activities and events. These lists of training activities can be linked to lists of resource requirements for training activities. These lists will reduce data entry requirements by allowing users to select items from a list in lieu of entering new data. Ideally, the system will also generate reminders to program operators about those training activities that should be periodically trained. These reminders may be generated from an adjunct system that records or predicts training proficiency over time.

Another important input/output feature of an automated scheduling system is feedback to users regarding scheduling failures. It will often be impossible to produce a set of Training Schedules that completely accommodates a highly constrained problem. The user needs to be informed of scheduling failures and the reasons for each failure so that appropriate action can be taken. For example, it may be impossible to schedule a Field Training Exercise due to the shortage of a renewable resource like MILES equipment. This shortage is potentially solvable by reallocating the timing of events. On the other hand, the shortage of an expendable resource like fuel is not solvable unless additional expendable resources are obtained. If users are informed of the reasons that events remained unscheduled, they may be able to take appropriate action to improve schedules (e.g., increase resources, or modify priorities).

The most basic reports required by an automated scheduling system are the Long- and Short-Range Calendars and Training Schedules. Updated Calendars and Training Schedules should be produced and distributed as modifications are made and approved. Additional reports that highlight <u>changes</u> in scheduled activities and <u>changes</u> in resource requirements may be particularly helpful. For example, if a renewable resource (e.g., classroom) is no longer required by one unit at a particular time, a report could highlight this fact, noting

new availabilities of renewable resources. Reports related to resource requests (e.g. ammunition) and resource utilization could also be produced (see ADEA, 1985). For renewable resources, it may be helpful for report resource usage as a function of unit and time.

Of particular interest at higher echelons in the Army is the issue of training cost. Reports can be extended to cover training cost as a function of resource utilization and resource cost. Such reports would require appropriate "cost models" that take into account the type of unit, the unit training location and additional relevant factors. Such training cost models would be based, at least in part, on Training Calendars and Schedules. For this reason, the automated scheduling work reported here should contribute to the automation of reports related to training cost.

USE OF SIMULATED ANNEALING TO SOLVE THE SCHEDULING PROBLEM

Alternative Approaches

In the operations research literature, linear programming is a method that is commonly employed to solve optimization problems for organizations. Optimization problems in organizations are varied. For example, a business may wish to identify the levels and kinds of inventory that maximize profit, or combinations of inputs that maximize a measure of efficiency, etc. Integer programming is a variant of linear programming that is applied to standard scheduling problems (Garfinkel & Nemhauser, 1972). Integer programming uses a solution vector restricted to integers while linear programming uses a continuous solution vector. Integer programming is applied to standard scheduling problems since the decision is a dichotomous (schedule/not schedule) one, which is a special case of integer programming. Integer programming problems can be solved by sequential linear programming runs in which improved optimizations to an integer solution are made at each step. The iterative nature of integer programming makes it a less efficient approach than linear programming, while integer programming is not necessarily too inefficient to solve Army scheduling problems, efficiency is a concern for large problems.

The primary drawback to the use of standard integer programming methods has to do with the heterogeneous nature of the Army scheduling problem. The variety of constraints involved in the Army scheduling problem means that a classical integer programming solution does not exist. In other words, an integer programming solution will not be able to handle simultaneously all of the types of constraints required by the Army problem and consequently will ignore some of them. In an Army operational environment, it is likely to be more problematic to obtain an "optimal" solution that ignores some essential Army constraints, than to obtain a good but not "optimal" solution that handles all of the essential types of constraints. Simulated annealing was selected as a useful candidate methodology because it can handle all of the essential types of constraints and still provide a "good" solution.

Description of Simulated Annealing

A methodology for approaching optimization is needed to solve the Army training schedule problem. Such a methodology provides (a) a definition of what constitutes a good schedule, (b) a way of measuring numerically how good a schedule is, and (c) a way to search to find numerically good schedules. Simulated annealing (Kirkpatrick, Gelatt & Vecchi, 1983) was selected as an appropriate method because of the match between the capabilities of the methodology and the characteristics of the scheduling problem. Simulated annealing operates by analogy to the metalurgy process which strengthens metals through successive heating and cooling. The method is highly dependent upon stocastic processes. In applying simulated annealing to scheduling, the concept of temperature is defined in terms of a quantitative objective function, termed a cost function. The approach is promising with respect to the Army's unit training scheduling problem because it can handle large problems. The training scheduling problem for an entire Army division is large. In addition, it can handle virtually any type of constraint. This feature is important because the Army training scheduling problem includes a wide variety of constraints. The approach is flexible in the sense that constraints can be changed or added and the method can still work. Flexibility is important because frequent modification is expected in an operational environment.

Another feature of simulated annealing that matches the Army training scheduling problem is size of the solution space. Application of simulated annealing is appropriate when multiple "good" solutions are acceptable as opposed to a uniquely "best" solution. The training scheduling problem matches this description. There is a relatively large class of "good" schedules and the goal is to find one of the schedules in this class. The solution space can be visualized geometrically as ridges and valleys, with multiple good solutions found along the valleys. Using this analogy increasing height corresponds to increasing cost or disutility. The "goal" is to not necessarily find the lowest point but a solution in a valley. The "best" possible solution is not likely to be much better than other solutions in the "good" class.

Further, for large combinational problems like the scheduling problem, it becomes impossible to check all possible combinations, in search of the one combination that minimizes the cost function, because of time requirements. The time requirements for such problems increase exponentially with <u>N</u> (i.e., number of activities to be scheduled). Rather than search all combinations, optimization methods use heuristics to minimize cost functions. Such heuristics can be generally classified as either "divide-and-conquer" or iterative improvement strategies (Kirkpatrick, Gelatt & Becchi, 1983). "Divide-and-conquer" or decomposition strategies partition a large problem into smaller ones that can be solved separately. The separate solutions then must be recomposed together to provide an overall solution. This approach is most appropriate when sub-problems are naturally disjoint so that the sub-problem solutions can be generated and then combined to form an overall solution without excessive distortion. For iterative improvement, a standard rearrangement operation is applied until a combination is found that improves the cost function. Typically, the rearranged configuration then becomes the

new configuration. The standard rearrangement operation is repeated until no further improvement in the cost function can be found. This type of search may get stuck in a local but not global minimum. For this reason, the iterative improvement search is often repeated using different random starting points. The simulated annealing method contains characteristics of both "divide-and-conquer" and iterative improvement strategies.

Finally, the simulated annealing algorithm does not require a set computer time to obtain a solution. The solution will get better on the average the longer the program runs. A computer run can be terminated when a point of diminishing returns is observed.

The simulated annealing heuristic operates by analogy to annealing in physical systems. Annealing in a physical system involves repeated heating and cooling of liquids or solids. When a liquid is hot there is a rapid random movement of the molecules making up the liquid. Heating a liquid increases the rapidity of the random movement of molecules. By contrast, cooling a liquid slows the random movement of molecules until they reach the point where they are frozen in place. Molecules frozen in place are considered "cold."

As applied to a scheduling system, the molecules are activities or events that must be scheduled. The concept of temperature is linked to the cost function, and refers to the degree of random assignments of activities to time slots. High temperature corresponds to extensive random assignment of activities to times. High temperature is associated with high values of the cost function, low temperature is associated with low values of the cost function. High temperature means that random assignment of activities to times is permitted no matter how large the cost function gets as a result of the assignments. On the other hand, cold temperature corresponds to minimal random assignments of activities to times, namely, assignments that reduce or at least do not increase the cost function.

The operation of the simulated annealing heuristic involves successively "heating" and "cooling" the system in search of a "good" schedule defined by a low cost function. The process of successively "heating" and "cooling" the system is accomplished to avoid the problem of getting stuck in local minimums. The problem of getting stuck in local minimums is a common one for problems based on iterative improvement (e.g., greedy algorithms). The simulated annealing heuristic requires the creation of an appropriate "annealing schedule" involving the extent and frequency of heating and cooling. An advantage of the simulated annealing heuristic is that annealing schedules help overcome the local minimums problem.

In addition to creating an appropriate annealing schedule, an algorithm is necessary to cool the system. (Creating a hot system is not difficult since heating simply entails unconstrained random assignment of activities to times.). Cooling the system, however, requires development of an algorithm, created especially for the problem at hand, that constrains the random assignments permitted to those producing successively lower values of the cost function, as the system is cooled. In order to "cool" a scheduling system, those tasks that have the greatest potential for increasing the cost function

are scheduled first before the schedule gets too full to include them. Then tasks that have less potential for increasing the cost function are successively scheduled. Unconstrained tasks that minimally add to the cost function are scheduled last. This strategy increases the probability that important activities and constraints are accommodated in the resultant schedules. The above ordered scheduling builds a "divide-and-conquer" strategy into the cooling algorithm. That is, the overall problem of scheduling is partitioned into a series of smaller sub-problems requiring scheduling sub-sets of activities in order of their importance.

Illustrative Application of Simulated Annealing

A simulated annealing heuristic approach to the Army training scheduling problem was implemented in a prototype test program to explore its feasibility. The test program was evaluated according to (a) computer efficiency (e.g., time requirements, memory requirements), (b) the face validity of the training schedules that are produced, and (c) flexibility of the program to handle additional requirements. In addition, the prototype test program will then be used to "fine tune" the design of the heuristic in terms of the selection and operation of (a) the cost function, (b) annealing schedules, and (c) types of constraints.

Description of the Test Program. The Army scheduling problem was delimited by focusing on scheduling at the battalion level. This level was selected because it contains scheduling features from all echelons. If the annealing heuristic is effective at this level, one can assume that it can be applied at all levels with the appropriate modifications for each echelon.

All five companies within a single battalion are essentially scheduled simultaneously in the test program. These five companies inherit training activities and events from the Long- and Short-Range Calendar created at higher echelons. The test program uses one-hour units of time for activities and forty-hour weeks. A final program will require smaller time units and the potential for longer weeks. The test program employs a very simple annealing schedule. Activities are assigned importance weights for decreasing "temperature" values (using the simulated annealing analogy). These weights, listed in order from high to low importance are based upon whether they: (1) appear as training activities on the Short-range Calendar, (2) require resources, (3) have high commander priority, (4) have a temporal relationship, or (5) have none of these characteristics, respectively. Conveniently, these same numerical assignments serve as the basis for the cost function, described in detail below. The system effects "cooling" by attempting to schedule activities with high values first.

The test program employs all of the important categories of scheduling constraints: (a) inheritance of activities from higher echelons and "fixing" activity times, (b) interschedule conflicts of both renewable and expendable resources, (c) company-level training activity priorities, (d) different unit priorities assigned to different companies, and (e) intra-schedule conflicts based on temporal constraints for activities (i.e., before/after, immediately before/immediately after, and temporal ordering of sequences of contiguous activities). The constraints that were considered most important were

included in the test program. Any exclusions were simply due to limitations on time and resources, and can be added as required for an applications version of the program.

The simulated annealing test program was written in FORTRAN 77 on a VAX 11/780 computer. The FORTRAN code for the test program is provided at Appendix A. An overview of the data structures and flow of control of the simulated annealing test program is shown in Figure 2. Blocks A through E in Figure 2 depict five data structures containing the necessary input information. An example of data input files and output schedules for an example problem is provided in Appendix B and is discussed later in this section.

It should be emphasized that the input files and output schedules do not represent files as they would be seen by users in a final scheduling program. Users of an implementation version will use input/output presented by a relational DBMS. The DBMS input/output will interface with the FORTRAN code that implements the simulated annealing heuristic.[2]

Initial Company-level Training Schedules developed by company commanders are represented in simplified form in Blocks F, G, and H of Figure 2. These schedules identify tasks for the week, assign priorities to the tasks ("high" versus "regular" priority; priorities may vary across companies), and place activities in temporal order consistent with prerequisite relationships (before/after). These initial schedules represent recommendations from a lower echelon for training schedules that are passed upward (See Figure 1). At battalion level, resourcing is accomplished and conflicts are resolved. As depicted in Figure 2, this conflict resolution is accomplished by the passage of the initially recommended schedules to the simulated annealing algorithm in Block I. The simulated annealing algorithm produces the final set of training schedules for the companies within a battalion. In addition, lists of activities from the initially recommended schedules that could not be accommodated in the final schedules are given along with their level of importance as defined by their contribution to the cost function.

---

[2]An important human factors problem was identified in previous automated scheduling research (see Medeiros & Yang, 1983a, 1983b) which will be corrected in future work. Figure 2 depicts five separate input files containing lists of activities and/or resources. These files were separated because they are conceptually distinct, and have parsimonious structures separately. Different files contain common subsets of activities and/or resources. In previous scheduling programs, users were required to enter the same items more than once in separate files. This procedure frequently produced data entry errors, due to heavy recall requirements for users who had to remember previously entered lists, and spelling variations of identical list items. These problems will be removed by having lists stored in some common data base to the extent possible. In addition, lists that are entered would only be typed in once by the user. Input would include automatic retrieval of previously entered lists for subsequent use in the context of different files, avoiding duplicate entry errors.

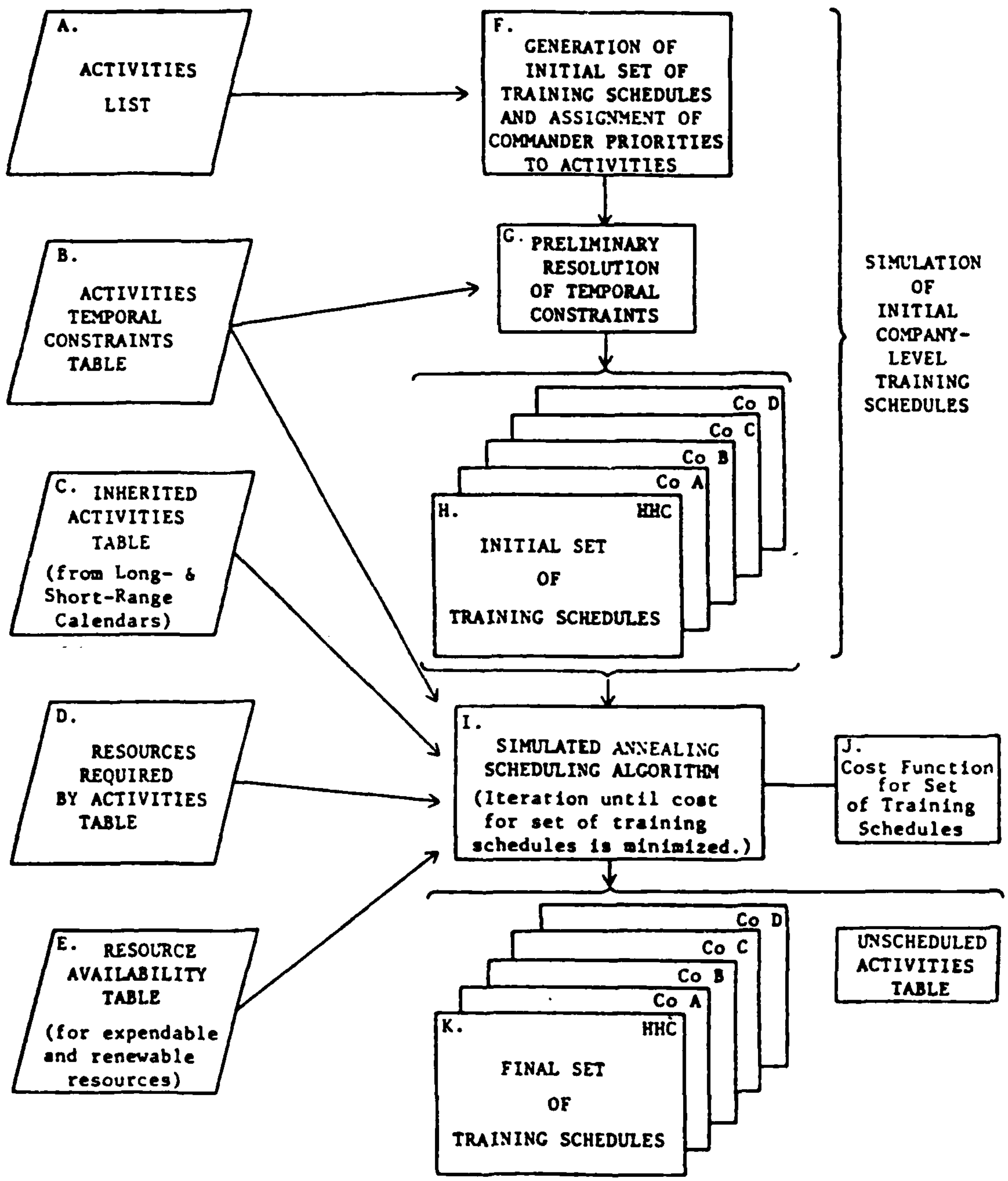


Figure 2. Data Structures and Flow of Control for Simulated Annealing Test Program

The cost function for the test program was formulated on the assumption that it is costly to fail to schedule activities. Further, it is more costly to fail to schedule longer than shorter activities. Thi is an idealization.

Specifically, let C represent cost and K represent weights reflecting the importance category (originated at higher echelon, requiring resources, company command priority, having temporal relationship, or regular activity) of unscheduled activities. Let U represent the time associated with unscheduled activities. Let j represent the activity number among m unscheduled activities within a Training Schedule, and let i represent the Training Schedule number that varies from 1 to n, the number of Training Schedules. Since there is a Training Schedule for each company, there are five Training Schedules per battalion ($n = 5$). Given these definitions, the cost function can be written by Equation 1 as:

$$C = \sum_{j=1}^{n} \sum_{i=1}^{m} K_{ij} \; U_{ij}$$

The simulated annealing algorithm, depicted in Block I of Figure 2, iteratively produces candidate sets of schedules. It employs the cost function to evaluate the schedule sets produced. As described below, when the simulated annealing algorithm is unable to generate substantially better schedules, it outputs both the Final Schedule Set, shown in Block K and a table of activities that could not be scheduled.

Figure 3 details the working of the simulated annealing scheduling algorithm. A general description of each of the primary operations within the algorithm is given below. The reader is directed to Appendix A for the computer code. The algorithm takes as input the Initial Company-level Training Schedules (and the data tables, Blocks A through F, shown in Figure 2). First, in Block 1 of Figure 3, some temporary storage space is cleared and other programmatic "housekeeping" is accomplished.

In Block 2 of Figure 3 the "cooling" process begins with the highest temperature, or most-costly-to-leave-unscheduled activities, which are those designated on the Short-range Calendar. Activities are taken in the order they appear on the calendar, implying no differential importance among them. Activities may or may not have a specified day and time for execution prescribed by the Short-range Calendar. If an activity does not have a specified time, first, the system checks to see if the activity already appears on the Initial Schedule for the company. If so, the activity is locked in the schedule and the system goes on to the next activity on the Short-range Calendar. When an activity is locked, it can no longer be moved or arbitrarily removed from the schedule by subsequent scheduler operations. When an activity does not appear in the Initial Schedule for the specified unit, and a specified time is not given, the system assigns the activity to a randomly selected time and locks it. In the case where a specified beginning time exists on the Short-range Calendar, the system checks whether the activity already is present in the schedule. If it is, it may be at the

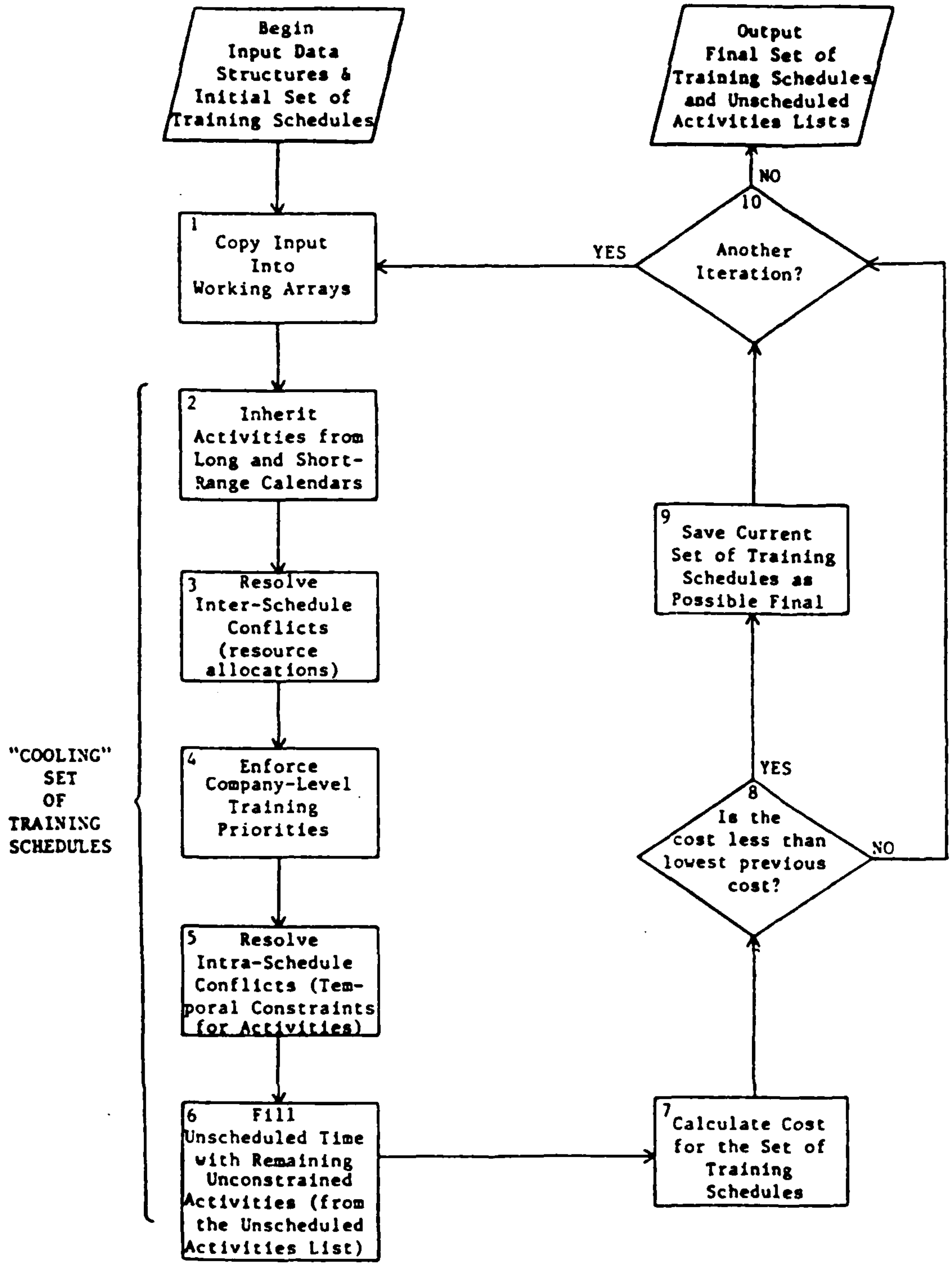


Figure 3. Operation of the Simulated Annealing Scheduling Algorithm

correct time or not. If it is not at the correct time it is moved. If it is not in the schedule it is inserted and, in any case, locked. Any displaced activities or conflicts among Short-range Calendar activities, such as two being erroneously assigned to the same start time for the same company, form the Unscheduled Activities Table.

The algorithm (Block 3) then considers the next "hottest" group of activities; those which require resources and, therefore, can involve inter-schedule conflicts. The interim schedules, (i.e., the Initial Schedules undergoing modification), are searched for activities which require resources. For each such activity, the Resource Availability Table is checked. If the resource required is expendable and available, the amount remaining is reduced by the required amount and the activity is locked. If the resource needed is a renewable resource, the system checks whether it is available at the required time. If so, the activity is locked and adjustments are made to the availability table. If a resource is not available at the needed time, the algorithm tries to exchange the time of the activity with other unlocked activities where the resource is available. If the activity can be moved to a time when the resource is available, it is locked there. If resources are not available for an activity, that activity gets added to the Unscheduled Activities Table.

In Block 4 of Figure 3, the algorithm works with those activities designated as Company-level Training Priorities. Each such activity already appearing in a schedule is locked. Those that appear on the Unscheduled Activity Table of the unit, unless they failed to be scheduled due to resource availability, are assigned a random time in the schedule and locked, but never displacing a locked activity.

Next, in Block 5 the algorithm works with those activities having temporal relationships which produce intra-schedule conflicts. Such activities, not previously locked, are moved around in the schedule; prerequisites are moved or added until all temporal relationships are satisfied and the corresponding activities are locked. Basically, the algorithm conducts an exhaustive search for situations which meet the temporal relationship of the activity being processed. Failure to accommodate a temporal relationship causes that activity to be removed from the schedule and added to the company's Unscheduled Activities Table.

Finally, the algorithm searchs for any unused time (Block 6) in the schedules and fills it with "low-temperature" activities on the Unscheduled Activity Table. The cost of the resulting schedule set is calculated (Block 7) and if the set is the best generated so far (Block 8) it is saved (Block 9), otherwise it is discarded.

After completing an iteration of the simulated annealing scheduling process the algorithm decides (Block 10, Figure 3) whether to repeat the process again from the Initial Training Schedules and data structures or not. Criteria for this decision are discussed below but basically involve specification by the user of a number of iterations where significant improvement in schedule quality ceases. If another iteration is to be executed, system control passes again to Block 1 of Figure 3 and the Initial

Set of Schedules is again "cooled." If not, then the best schedule set and corresponding Unscheduled Activities Table is output and the system terminates.

Operation of the Test Program. Appendix B shows a sample run with complete training schedules for each of five companies at each successive stage of one iteration. To illustrate the types of schedule manipulations the system produces, Table 1 shows a segment extracted for Company B on Monday at each of the six stages in one "cooling" iteration.

The first column of Table 1 shows the initial schedule for Company B produced by randomly sampling from the population of 60 activities without replacement for each of the five companies. Three activities were randomly selected as company-level training priorities and are marked with an asterisk. This initial schedule can be considered the company-level proposed training schedule that is forwarded to battalion for verification and approval.

The second column shows the schedule segment after activities have been inherited from the higher echelon calendars. As shown in Table 1, two activities appear at the required hours designated by the Long- and Short-term schedules: PT and its necessary immediate successor, PERS HYGIENE. Other activities from the initial schedule are inserted either at their designated times, or, if times are not specified, at randomly selected times. (No activities with unspecified times occurred in this schedule segment). All inherited activities are fixed in the schedule for the duration of the iteration. Activities with designated times will not be moved but may be removed from the schedule due to resource or temporal constraint violations. Those with randomly chosen times may be moved or removed. The activities that are removed are placed on the initial unscheduled activities list for the company being processed. In this segment, MISSION SUPPORT and ARCRFT REC 1 went on the unscheduled activity list for Company B.

The third column of Table 1 shows the results of the resource allocation step. FIRST AID 4 requires resource INST MOE which is available at the time the activity has been scheduled. ID TERR FEAT needs resource TR ARA 2 which is not available at 1500. In fact, it was initially available at this time, but, (as shown in Appendix B), this resource was allocated at 1500 to Company A, which has a higher unit priority. Therefore, ID TERR FEAT is exchanged for AREA MAINT 4, and a time is found at which the needed resource is available. Similarly, activity INSTALL M18 requires TR ARA 3 which is unavailable on Monday and the activity is moved to a time on another schedule (not shown) when the resource is available. If a required resource were unavailable at any time, the activity would be added to the unscheduled activity table. When resource allocations have been completed, all activities requiring explicitly allocated resources become fixed in the schedule, although they may still be removed for temporal constraint violations.

At the fourth step, activities designated as company-level priorities are fixed in the schedule. If priority activities have been previously scheduled (such as NERVE AGNT3 in Table 1), they are fixed in the schedule. If priority activities are not currently scheduled, they are found on the unscheduled

Table 1

Illustration of Schedular Functioning for Monday, Company B (from Appendix B)

	Stage of algorithm					
Time	Initial schedule	Inheritance	Resource allocation	Company-level priorities	Temporal relationships	Final schedule
8:00	MISSION SUPP*	PT	PT	PT	PT	PT
9:00	ARCRFT REC 1*	PERS HYGIENE	PERS HYGIENE	PERS HYGIENE	PERS HYGIENE	PERS HYGIENE
10:00	FIRST AID 4	FIRST AID 4	FIRST AID 4	FIRST AID 4	FIRST AID 4	FIRST AID 4
11:00	INSPECTION 2	INSPECTION 2	INSPECTION 2	INSPECTION 2	NERVE AGNT 1	NERVE AGNT 1
13:00	INSPECTION 3	INSPECTION 3	INSPECTION 3	INSPECTION 3	INSPECTION 3	INSPECTION 3
14:00	NERVE AGNT 3*	NERVE AGNT 3	NERVE AGNT 3	NERVE AGNT 3	NERVE AGNT 3	NERVE AGNT 3
15:00	ID TERR FEAT	ID TERR FEAT	AREA MAINT 4	MISSION SUPP	MISSION SUPP	MISSION SUPP
16:00	INSTALL M18	INSTALL M18	PREP H72 1	PREP H72 1	MAINT M16 1	MAINT M16 1

*Activities designated as company-level training priorities.

activities list. These activities are then, if possible, substituted for activities that have not yet been fixed in the schedule. In the current implementation, only unscheduled priority activities not requiring resources are rescheduled. These priority activities are then fixed so that they cannot be removed during this iteration. As shown in column four of Table 1, MISSION SUPP, which had been removed earlier, is rescheduled due to commander priority. The priority activity ARCRAFT REC 1 is rescheduled on Friday (see Appendix B). Activities that are displaced (in this case, AREA MAINT 4) are added to the unscheduled activities list.

In step 5, temporal relationships are resolved. PERS HYGIENE which was placed in the schedule at the inheritance step is verified as an immediate successor to PT. If it had not been present, an attempt would have been made to add it. The activity MAINT M16 2 which appears later on Tuesday in the Company B schedule requires the prerequisite MAINT M16 1. The time slot with an unfixed activity is on Monday at 1600 so the prerequisite MAINT M16 1 is inserted. Again, the displaced activity, as well as any activities with temporal relationships that cannot be satisfied, are added to the unscheduled activities table. Finally, if any unscheduled blocks of time exist in the schedule, activities would be selected from the unscheduled activities table and put into the schedule. No such change occurred in the particular schedule segment in Table 1. Thus, final schedule shown in column 6 of Table 1, is identical to column 5.

After the final schedule has been determined, the cost is computed. As described earlier, the cost is based on the importance and duration of the unscheduled activities. If an activity has more than one type of constraint, it can be assigned more than one level of importance when the activity is unscheduled. In this circumstance, the highest importance weight is assigned to the activity. Costs are computed for each unscheduled activity and summed across unscheduled activities for a company. Then company costs are summed across all companies in a battalion, yielding a total cost value for the schedule set (see Equation 1 and Subroutine Cost in Appendix A).

In an application version of such a program, normally only the final schedule set would be of interest and would be printed. The application programs would also provide the unscheduled activities list, along with the specific reasons for exclusion, so that manual modifications of the schedule could be made by users as they deemed necessary. The purpose here has been to provide insight to the reader of the logic and inner workings of the current developmental scheduling system.

Evaluation of Test Program. To evaluate the simulated annealing test program, hypothetical scheduling problems were devised by creating the five required data entry tables shown in Figure 2 (labelled A-E). The "easy" scheduling problem had (a) an average of four activities per company inherited from the higher echelon calendars, (b) a total of ten activities (from the 60 possible) which required one or more of four different resources, (c) ten percent of the initially scheduled activities designated as company-level commander training priorities, and (d) five activities with some type of temporal relationship to another activity. The "difficult" scheduling problem had: (a) an average of 12 inherited activities per company, (b) 30 activities

requiring one or more of 12 types of resources, (c) 20 percent of initially scheduled activities designated as priorities, and (d) 15 activities in temporal relationships with other activities. (The complete data tables defining the difficult problem may be found at the end of Appendix B). Final sets of schedules for the five companies of a battalion produced by the test program for these problems appear to be satisfactory in quality and face validity although a formal evaluation from subject matter experts may be required for further substantiation.

Figure 4 presents results showing the reduction in cost functions for problems as the number of iterations through the "cooling" mode increases. Results are presented separately for an "easy" and "difficult" scheduling problem. The minimum cost function prior to the specified iteration is provided. In Figure 4 the minimum value was averaged over ten cases (i.e., ten easy problems, ten difficult problems). The cost values, which were larger in an absolute sense for the difficult problem, have for both problems been mapped onto the zero to one interval for comparison. As the improvement in schedule set quality was negligible after the seventieth iteration, the graph is truncated.

An important finding shown in Figure 4 is the rapid rate of schedule set improvement, as illustrated by the rapid decline of the cost function. The improvement in the cost value reaches its asymptotic limit quite quickly, well before 100 iterations. This trend suggests that the current implementation is relatively efficient, at least as indicated by the number of iterations necessary to achieve a stable solution. A second finding is the very similar rate of decrease of the cost function for both easy and hard problems.

Run-time statistics were also examined to further investigate the performance characteristics of the test program. Running on a VAX 11/780 for 100 iterations, the easy problem required an average of about four minutes of CPU time. The difficult problem required about 5.7 minutes of CPU time. Wall clock time was about ten percent more when saving only the best schedule set at any given point. These problems used a maximum of 98,000 bytes of memory, including data tables.

Given the above statistics, it may be feasible to run scheduling problems of this size on microcomputers. However, "overnight" runs would probably be required. On one hand, an operational program could make more efficient use of computer resources than the test program. In addition, it is unlikely that real problems would require 100 iterations. On the other hand, actual Army scheduling in practice may produce much larger data tables which could be so voluminous that it might be impossible to enter all data in the memory of a microcomputer at one time. This situation would slow the program operation down considerably.

CONCLUSIONS AND RECOMMENDATIONS

The simulated annealing heuristic approach to the Army scheduling problem, as implemented here, appears promising in several respects. First, it seems to effectively accommodate the highly heterogeneous and unique aspects of the problem. The ease with which the approach fits into the hierarchical

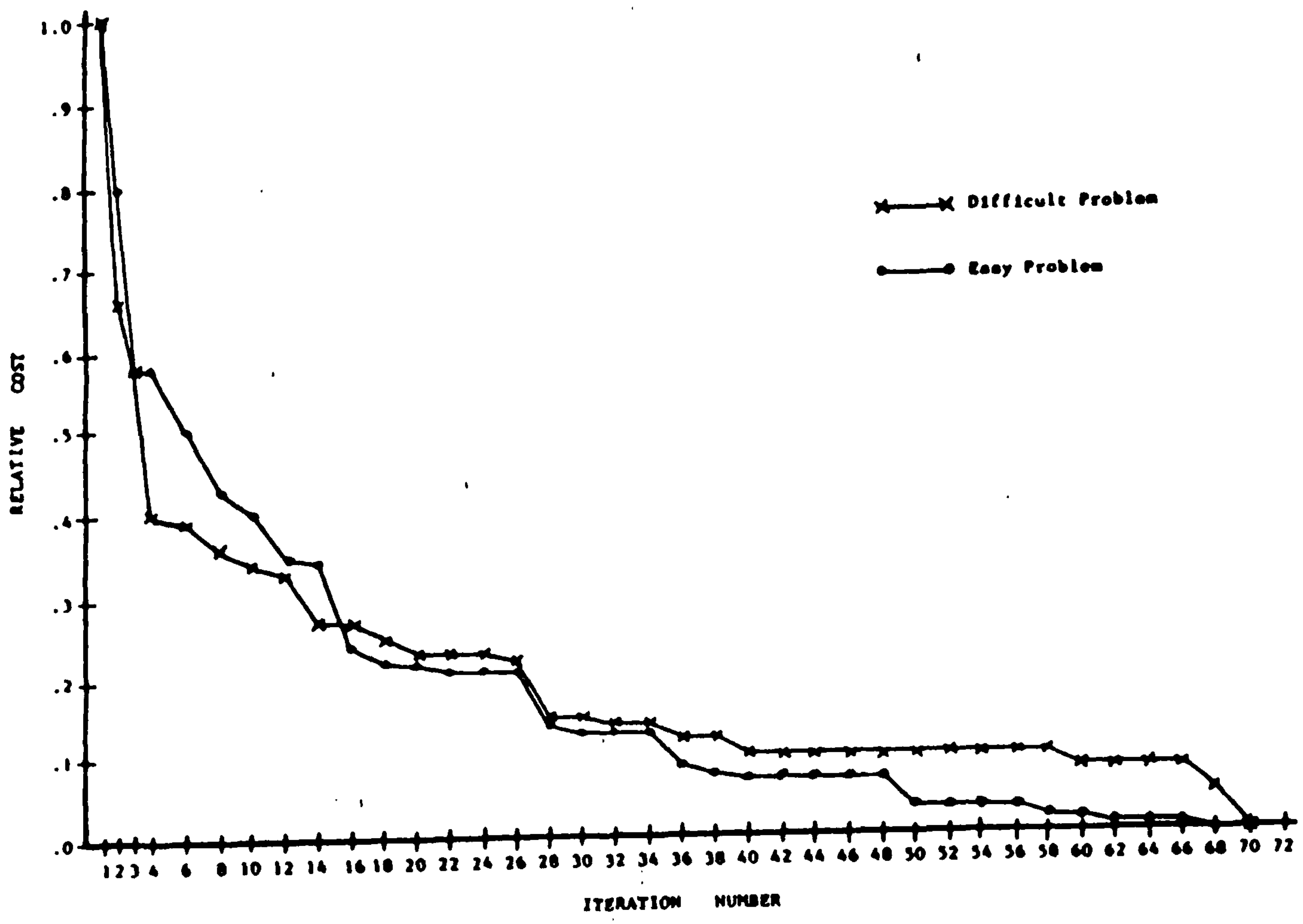


Figure 4. Relative cost by iteration for best schedule sets using easy and difficult scheduling problems.

organizational structure of the Army suggests that the same principles, and perhaps actual sections of code, could be adapted to scheduling of Long- and Short-Range Calendars at brigade or division level. In addition, difficult scheduling problems appear to be handled rather efficiently with minimal increases in computer resource requirements. As such, we believe that the general approach is worthy of continued development.

On the other hand, the existing program has serious limitations when viewed from the perspective of what will ultimately be required in an applications program. The test program incorporated simplifying assumptions such as using hours as the smallest time unit, scheduling one week at a time, and tacitly assuming that a company has only one activity at any given time. Further, certain constraints were imposed upon the nature of the input data. For example, activities on the right side of a temporal relationship (i.e., constrained to occur after another activity) were assumed not to require resources or themselves to appear on the left side of a temporal relationship (i.e., constrained to occur before another activity). None of these simplifications appear to be unchangeable. Rather, they have been, to date, merely temporary limitations that were subordinate to the goal of testing the overall feasibility of the method. Modifications to incorporate additional complexities entail both development time and probably some code expansion. A potential limitation of the approach is that the final schedule set produced typically will not be optimal in a global sense. The cost function goes down relatively quickly, but the "good" schedules as defined by a low cost function, may still contain clearly non-optimal assignments of activities.

Given that the general method of simulated annealing shows promise for solving or at least contributing to a solution to the Army's unique scheduling problems, several development directions for improving schedules have become apparent to us as a result of the work to date. One such modification which appears likely to yield improvements in schedule quality involves allowing variation in annealing schedules. This variation would permit local small-scale heating and cooling. For example, in the current version of the test program, once resources have been allocated, no further resource allocation occurs for the duration of the iteration. If local variation in the annealing schedules were permitted, it would be possible to reassign resources after temporal constraints are met in a local iterative loop. It may also be productive to explore other ways of having the successive iterations build directly upon the output of the previous iterations rather than beginning completely anew each time, as is now the case.

Another possibility for improving performance might involve reconceptualizing the concept of "temperature." Instead of defining temperature based on stages involving different types of constraints, as is now the case, temperature could be defined based on the temperature of activities. Some activities could be considered "hotter" than others based on the number of constraints imposed on them. Thus, activities with a greater number of constraints have a larger number of potential constraint violations and would be scheduled first. Such a redefinition of temperature would constitute a radical modification of the existing system but appears to be worth pursuing.

Another area for future development that would support the software development effort involves evaluation of Army Training Schedules. This effort might further explore what constitutes a realistic Army Training Schedule. The definitions of easy and difficult problems used here are useful for exploring the strengths and weaknesses of the simulated annealing method. However, they are somewhat arbitrary in terms of the specifics of actual Army scheduling. There is a need for evaluation of the quality of schedule set output based on its utility to those charged with the actual planning of training. It is important to determine whether the scheduling outputs from the test program are satisfactory and whether they provide sufficient benefits to overcome the costs incurred in their generation.

REFERENCES

Army Development and Employment Agency (June 1985). Functional description for the Integrated Training Management System (ITMS) (Vol. I). Ft Lewis, WA: BDM Corporation.

Garfinkel, R. S. & Nemhauser, G. L. (1972). Integer Programming. New York, Wiley.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. Science, 220, 671-680.

Medeiros, D. J. & Yang, T. (June 1983a). Base scheduling and resource estimation model (Contract No. DABT60-80-C-0049). University Park, Penn: Pennsylvania State University.

Medeiros, D. J. & Yang, T. (December 1983b). Base scheduling and resource estimation model (Contract No. DABT60-80-C-0049). University Park, Penn: Pennsylvania State University.

Van der Eijk, J. A., Ignizio, J. P., & Yang, T. (1981). Base scheduling and resource estimation for the Army Battalion Training Model (Vol. 3), (Contract No. DABT60-80-C-0049). University Park, Penn.: Pennsylvania State University.

Yang, T. & Ignizio, J. P. (1982). Base scheduling and resource estimation for the Army Battalion Training Model (Contract No. DABT60-80-C-0049). University Park, Penn: Pennsylvania State University.

APPENDIX A

COMPUTER PROGRAM LISTING

```
C======================================================================
C
C            This is the main calling program for
C            the implementation of the simulated
C            annealing algorithm to Army scheduling.
C            14 June 1985,   ARI-POM
C
C======================================================================
C
        CHARACTER IN*1
C
        INCLUDE 'COMMON.FOR/LIST'    !common data block
C
        CALL SCHBLD            !build the schedules
        CALL PRCD(SCHA_I)      !check temporal relationships
C
        WRITE(5,10)
10      FORMAT(/'$How many iterations would you like?')
C
        READ(5,20) ITERS
20      FORMAT(I3)
        WRITE(5,11)
11      FORMAT(/'$Would you like all iterations saved to disk?')
        READ(5,12) IN
12      FORMAT(A1)
        RECORD=0
        IF( (IN(1:1).EQ.'Y').OR.(IN(1:1).EQ.'y')) RECORD=1
C
        DO I=1,ITERS     ! iteration loop
C
                CALL ANNEAL(ICOSTMIN,I,RECORD)
C
                WRITE(5,505) I
505             FORMAT(/' Iteration number = ',I3)
        ENDDO
C
C write out the minimum cost
C
        WRITE(5,25) ITERS,ICOSTMIN
25      FORMAT(//' Number of iterations = ',I4,/,' Minimum cost = ',I4)
C
C
C let's save the best set
C
        OPEN (UNIT=8,FILE='BESTSET.DAT',STATUS='NEW')
C
        CALL PRTSCH2(A,SCHA_S)
        CALL PRTFAIL2(FAILA_S,FAILPRA_S)
        CALL PRTSCH2(B,SCHB_S)
        CALL PRTFAIL2(FAILB_S,FAILPRB_S)
        CALL PRTSCH2(C,SCHC_S)
        CALL PRTFAIL2(FAILC_S,FAILPRC_S)
        CALL PRTSCH2(D,SCHD_S)
        CALL PRTFAIL2(FAILD_S,FAILPRD_S)
        CALL PRTSCH2(HHC,SCHH_S)
        CALL PRTFAIL2(FAILH_S,FAILPRH_S)
C
C
        WRITE(5,30)
30      FORMAT (/' Bye')
        END              ! all done, that's all there is to it
```

```
C
        SUBROUTINE ANNEAL(ICOSTMIN, ITER, RECORD)
C===================================================================
C
C           This is the implementation of the
C
C           SIMULATED ANNEALING ALGORITHM
C
C           Created 22 MAY 1985    djg
C
C===================================================================
C
        CHARACTER HEAD*40
C
        INCLUDE 'COMMON.FOR/LIST'
C
        CALL INIT !clears arrays etc
                HEAD(1:40) ='*****  INITIAL SCHEDULE *****'
                IF (RECORD.EQ.1) CALL SAVEALL(HEAD,ITER)
        CALL SRC    ! short-range calendar
                HEAD(1:40)='***** SRC inherited *****'
                IF (RECORD.EQ.1) CALL SAVEALL(HEAD,ITER)
        CALL RSRC    ! resources
                HEAD(1:40)='***** RESOURCES CHECKED *****'
                IF (RECORD EQ 1)CALL SAVEALL (HEAD,ITER)
        CALL CDRPR ! puts CO. CDR priority activities on schedule
                HEAD(1:40)='***** COMMANDER PRIORITIES RESCHEDULED *****'
                IF (RECORD.EQ.1)CALL SAVEALL(HEAD,ITER)
        CALL TMPRL ! checks temporal relationships
                HEAD(1:40)='***** TEMPORAL RELATIONSHIPS CHECKED *****'
                IF (RECORD.EQ.1)CALL SAVEALL(HEAD,ITER)
        CALL UNSCH ! fills out the schedule
                HEAD(1:40)='***** UNSCHEDULED ACTIVITES--FINISHED *****'
                IF (RECORD.EQ.1)CALL SAVEALL(HEAD,ITER)
        CALL COST(ICOST) ! computes the cost of the schedule set
        CALL SAVE(ICOST,ICOSTMIN) 'decision to save results of this iteration
        RETURN
        END
C===================================================================
C       Common block
C===================================================================
C
C
C       SCHA_I is schedule for company A initial
C       SCHB_I is company B
C       SCHC_I is company C
C       SCHD_I is company D
C       SCHH_I is company HHC
C
C       SCHPRA is cdr priorities for Company A
C       SCHPRB for company B, etc.
C
C       SCHA is working schedule for company A
C       SCHB is for B, etc.
C
C       SCHA_S is saved schedule for Company A. SCHB_S for B, etc.
C
C       A,B,C,D,HHC are table titles.
C
C       FAILA,FAILB,etc are the fail to schedule lists
C          task (c12) and reason (c8)
```

```
C            max of 40 per unit
C            There sre associated FAILPR vectors which
C               give the cdr priority of the task.
C
C         FAILA_S.FAILB_S etc saves the best so far
C
C         FAILPR(41) is used as a pointer to next free line in the
C              fail arrays.
C
C         LOCKEDA.LOCKEDB.etc show whether a task has been locked
C            at a higher "temperature" level.
CAMAIN    This is the main program implementing the annealing algorithm
C         applied to the Army scheduling problem.
C
CAAAREADME.FOR  Is this documantation file.  Type DOC to read/edit.
C
CALIB.OLB  This is the .OBJ library.  To include or replace an item
C         use the command: CL <filename>.  Next you may: GO to
C         test the changes you've made.
C
CRN(N)  A function returning a random integer between 1 and N.  It uses
C         RND(I).  Returns zero if N=0.
C
CRND(I) The actual random number generater.  It creates its own seed
C         based on reading the system clock.
C
CPRTSCH(TITLE,SCH)  A utility subroutine that prints the title (char 40)
C         and schedule information (char 480) to the screen in calendar
C         format with an option of printing hardcopy.
C
CTSKLST.DAT  Data structure containing candidate training activities.  First
C         field of three numerials is serial id, next field of 12
C         characters is description  Activities are on even numbered file
C         lines, odds are used for typing guides.
C
CSCHBLD Subrountine that manages the building of schedules for all
C         companies.
C
CSCHBLDAX(SCHED,SCHPR)  Subroutine that builds initial schedules from the
C          TSKLST items using random selection.  Also assigns random
C          priorites to the tasks (range 1 to 40 with unique values).
C
CTSKSWAP(IHR1,IHR2,SCH,SCHPR)  In schedule SCH will exchange the task in IHR1
C          with that in IHR2.
C
CTSKSRCH(TSK,SCH,LOC,ISTART)  In schedule SCH will serach from hour
C          ISTART to end for task TSK (char 12) returning location hour LOC
C          or zero if not found.
C
CTSKRPLC(LOCATION,TASK,SCH)  Replaces task in schedule with TASK.
C
CPRCDTBL.DAT  Table of precedence relationships among tasks
C           Format:  TaskID (c12) relation (c6) TaskID (c12)
C          Current valid relationships:
C                 AFTER -- task1 must occur after another task2
C                           (1)  task2 will be swapped to an earlier time
C                           than task one if possible and already in the
C                           schedule or
C                           (2) it will be added if not in schedule or
C                          (3) a fail message will be generated
C
```

```
CSRC.DAT  The Short-range Calendar.  Maximum of 100 elements.
C         Task (c12)
C         Unit (c1) A,B,C,D,H,Every
C         Hour (i2), 1-40, 0 = none specified
C
CPRCD(SCH)  Subroutine which will check the table for precedence
C            violations and attempt to resolve them.
C
CPAFTER(TSK1,LOC,TSK2,SCH,LOCKED_HOUR_TABLE)
C            Subroutine deals with precedence relationahip
C            where TSK1 in location LOC
C            can appear only after TSK2 in schedule SCH.
C            It is called from PRCD.
C           Uses and updates table of tasks that are locked in terms
C            of precedence only.
C
CSRC     The subroutine which reads SRC.DAT and changes the company
C        schedules accordingly.  Shares COMMON.FOR
C
CSRCAUX  Subroutine that directly does the work on each company.  It
C        called only by SRC and shares a common block with it.
C
CSCHCOPY(INSCH,OUTSCH,INSCHPR,OUTSCHPR)  Utility for copying one schedule
C        into another.
C
CTSKRSRC.DAT  Table of resources required by tasks.  Contains tasks (c12)
C          and resources (c8).  Maximum of 50 lines.
C
CRSRC.DAT      Table of the actual resources available.  Contains the
C              resource name (c8),a quantitative amount available (i2)
C              for expendable resources, and a 40 hour calendar for
C              renewable resources (i40) where a 0 means the resource
C               is not available at that hour.  A maximum of 50 resources
C              can be entered.
C
CRSRC  Subroutine that resolves resource contraints and interunit
C        conflicts
C
C
CRSRCAUX(SCH,SCHPR,FAIL,FAILPR,LOCKED,TSKRS,RSLST,RSTIME,RSQTY)  Does the
C        actual work on each schedule for RSRC.
C
CHIGHPR(FAILPR,TRIED,IPNT)  Subroutine that returns the pointer to the
C        highest cdr priority (lowest value) on the fail list, or zero
C        if all have been tried.
C
CTSKSUB(sub,schpr,itarget,tskin,tskout,fail,failpr,ipnt,reason)
C        Subroutine that substitutes TSKIN in schedule SCh at hour
C        ITARGET, removing that task, TSKOUT, putting it on the fail
C        list with REASON>  Also swaps priorities.  If task in schedule
C        is blank, reduces size of fail list.
C
CTMPRL Subroutine which checks temporal relationships for each company
C       schedule.  It includes the common block.  Calls TMPRLAUX for each
C       company.
C
CTMPRLAUX(SCH,SCHPR,FAIL,FAILPR,LOCKED,PRC)  Performs the check on
C        temporal relationships for each company.  Either resolves the
C        dependency or pulls the task from the schedule and adds it
C        to the unscheduled (FAIL) list.
C
```

```
CREASON (LOCKED,IHR,REA)  Returns the level at which a scheduled activity
C       has been locked.
C
CUNSCH Puts low priority activites, requiring no resources or having no
C       temporal relationships on the schedule.
C
CUNSCHAUX Does the job for UNSCH on each company.
C
CPRTFAIL(fail,failpr)  Prints the fail list.
C
CINIT Clears buffers etc. at the start of each iteration
C
CSAVEALL(HEADING,ITERATION_NUMBER)  Saves everything onto disk for later
C       review.
C
CPRTDSK(LABEL,SCHEDULE)  Same as prtsch only to disk.
C
CLCKSETUP(TEMP,LOCKED)  Puts locked info into schedule format for
C       SAVEALL.
C
CPRISETUP(TEMP,PRIOTITY) Puts cdr priority info into schedule format for
C       use by SAVEALL.
C
C
CFAILCOPY  Copies fail lists into save array for best schedule set so far
C
C
CPRTSCH2 & PRTFAIL2  Output the best set of schedules for BESTSET.DAT.
C
C
        CHARACTER*480 SCHA_I,SCHB_I,SCHC_I,SCHD_I,SCHH_I,
     $     SCHA,SCHB,SCHC,SCHD,SCHH,
     $     SCHA_S,SCHB_S,SCHC_S,SCHD_S,SCHH_S
        COMMON SCHA_I,SCHB_I,SCHC_I,SCHD_I,SCHH_I,
     $     SCHA,SCHB,SCHC,SCHD,SCHH,
     $     SCHA_S,SCHB_S,SCHC_S,SCHD_S,SCHH_S
C
        CHARACTER*40  A,B,C,D,HHC
        COMMON        A,B,C,D,HHC
C
        CHARACTER*800 FAILA,FAILB,FAILC,FAILD,FAILH,
     $        FAILA_S,FAILB_S,FAILC_S,FAILD_S,FAILH_S
        INTEGER FAILPRA(41),FAILPRB(41),FAILPRC(41),FAILPRD(41),
     $         FAILPRH(41),SCHPRA(40),SCHPRB(40),SCHPRC(40),
     $         SCHPRD(40),SCHPRH(40),
     $         FAILPRA_S(41),FAILPRB_S(41),FAILPRC_S(41),FAILPRD_S(41),
     $         FAILPRH_S(41),SCHPRA_I(40),SCHPRB_I(40),SCHPRC_I(40),
     $         SCHPRD_I(40),SCHPRH_I(40)
        COMMON  FAILA,FAILB,FAILC,FAILD,FAILH,
     $         FAILA_S,FAILB_S,FAILC_S,FAILD_S,FAILH_S,
     $          FAILPRA,FAILPRB,FAILPRC,FAILPRD,FAILPRH,
     $          SCHPRA,SCHPRB,SCHPRC,
     $          SCHPRD,SCHPRH ,
     $         FAILPRA_S,FAILPRB_S,FAILPRC_S,FAILPRD_S,
     $         FAILPRH_S,SCHPRA_I,SCHPRB_I,SCHPRC_I,
     $         SCHPRD_I,SCHPRH_I
C
        COMMON LOCKEDA(40),LOCKEDB(40),LOCKEDC(40),LOCKEDD(40),LOCKEDH(40)
C
          A(1:40)='       SCHEDULE FOR COMPANY A'
          B(1:40)='       SCHEDULE FOR COMPANY B'
```

```
          C(1:40)='       SCHEDULE FOR COMPANY C
        D(1:40)='       SCHEDULE FOR COMPANY D'
        HHC(1:40)='       SCHEDULE FOR HHC    '
C
C End of common block
C=====================================================================
        SUBROUTINE CDRPR
C=====================================================================
C  substitutes tasks on the fail list with high cdr priorities for
C  tasks in each company's schedule which have lower priorities
C  15MAY85  djg
C
        INCLUDE 'COMMON FOR/LIST'
C
        WRITE(5,1)
1       FORMAT(/' ***** Resolving Company CDR Activity Priorities *****')
C
        CALL CDRPRAUX(SCHA,SCHPRA,FAILA,FAILPRA,LOCKEDA)
        CALL CDRPRAUX(SCHB,SCHPRB,FAILB,FAILPRB,LOCKEDB)
        CALL CDRPRAUX(SCHC,SCHPRC,FAILC,FAILPRC,LOCKEDC)
        CALL CDRPRAUX(SCHD,SCHPRD,FAILD,FAILPRD,LOCKEDD)
        CALL CDRPRAUX(SCHH,SCHPRH,FAILH,FAILPRH,LOCKEDH)
C
        RETURN
        END
C=====================================================================
        SUBROUTINE CDRPRAUX(SCH,SCHPR,FAIL,FAILPR,LOCKED)
C=====================================================================
C  does the work on each schedule regarding cdr priorities
C
        CHARACTER SCH*480,FAIL*800,TSKIN*12,TSKOUT*12,CP*8
        CHARACTER TSKRS*1000
        INTEGER SCHPR(40),FAILPR(41),LOCKED(40),TRIED(40)
C
C read in resource table first time only
        IF (IFLAG.NE.1) THEN
                IFLAG=1
                OPEN (UNIT=3,FILE='TSKRSRC.DAT',STATUS='OLD')
                DO I=1,50
                J=(I-1)*20+1
                READ(3,11) TSKRS(J:J+19)
11              FORMAT(/A20)
                ENDDO
        ENDIF
        CP(1:8)='TIME    '
C  clear tried vector
        DO I=1,40
          TRIED(I)=0
        ENDDO
3       CONTINUE
C lock all of the priority activities now in the table
C not already locked
        DO I=1,40
                IF((SCHPR(I).EQ.1).AND.(LOCKED(I).EQ.0)) LOCKED(I)=3
        ENDDO
        CALL HIGHPR(FAILPR,TRIED,IPNT)
C  here is the stop condition, when all have been tried
C
        IF (IPNT.EQ.0) RETURN
C
            TRIED(IPNT)=1
```

```
C  make sure the activity does not require any resources  29may
        DO I=1,50
                J=(I-1)*20+1
                IF(TSKRS(J:J+2).EQ.'END') GOTO 21 !done
                K=(IPNT-1)*20+1
                IF(FAIL(K:K+11).EQ.TSKRS(J:J+11)) GOTO 3 !needs resources
        ENDDO                                                     !get another
21      CONTINUE
C  make sure at least one task is in calendar and not locked, and
C  make sure task with a lower cdr priority exists in sch
                DO IHR=1,40
                IF ((LOCKED(IHR).EQ.0) .AND. (SCHPR(IHR).GT.FAILPR(IPNT)))
     $            THEN
C search for and use blanks first 29 may
                        DO I=1,40
                                IJPNT=(I-1)*12+1
                                IF(SCH(IJPNT:IJPNT+11) .EQ. '            ')
     $                            THEN
                                J=(IPNT-1)*20+1
                                TSKIN(1:12)=FAIL(J:J+11)
                                CALL TSKSUB(SCH,SCHPR,I,TSKIN,TSKOUT,
     $                            FAIL,FAILPR,IPNT,CP,TRIED)
                                LOCKED(I)=3
                                GOTO 3
                                ENDIF
                        ENDDO
10                ITARGET=RN(40)    !pick one at random
                  IF ((LOCKED(ITARGET).EQ.0) .AND.
     $                 (SCHPR(ITARGET).GT.FAILPR(IPNT)))   THEN
                        J=(IPNT-1)*20+1
                        TSKIN(1:12)=FAIL(J:J+11)
                        CALL TSKSUB(SCH,SCHPR,ITARGET,TSKIN,TSKOUT,
     $                    FAIL,FAILPR,IPNT,CP,TRIED)
                        LOCKED(ITARGET)=3
                        GOTO 20
                  ENDIF
                  GOTO 10  ! choose another target, this one no good
20                CONTINUE
                ENDIF
                ENDDO
        goto 3  !continue until all are tried
        END
C==================================================================
        SUBROUTINE COST (IC)
C==================================================================
C
C computes the cost for a set of schedules, returns total cost
C
        INCLUDE 'COMMON.FOR/LIST'
C
        CALL COSTAUX (FAILA,FAILPRA,ICA)
        CALL COSTAUX (FAILB,FAILPRB,ICB)
        CALL COSTAUX (FAILC,FAILPRC,ICC)
        CALL COSTAUX (FAILD,FAILPRD,ICD)
        CALL COSTAUX (FAILH,FAILPRH,ICH)

C
        IC=ICA+ICB+ICC+ICD+ICH
C
        WRITE(5,100) IC
100     FORMAT(/' Cost of this schedule set is ',I3,//)
```

```
      WRITE(2,101) IC
101   FORMAT(/' Cost of this schedule set is ',I3,//)
C
C
      RETURN
      END
C==================================================================
      SUBROUTINE COSTAUX(FAIL,FAILPR,ICOST)
C==================================================================
C
C this computes the cost of the schedule passed to it based on the fail list
C
      CHARACTER FAIL*800,REASON*8
      INTEGER FAILPR(41)  !remember 41 stores the next empty cell
C                          ! on the fail list
C for every item on the fail list compute its cost
C
      ICOST=0
      DO ITEM=1,FAILPR(41)-1
      IREASON=(ITEM-1)*20+13  !pointer into the fail array
      REASON(1:8)=FAIL(IREASON:IREASON+7)   !get the reason
C
C here we go looking at the specific reason.  Values ARE flexibile!
C
      IF    (REASON(1:8).EQ.'SRC     ') THEN
C
            ICOSTINC=5
C
      ELSEIF(REASON(1:8).EQ.'RESOURCE') THEN
C
            ICOSTINC=4
C
      ELSEIF(FAILPR(ITEM).EQ 1) THEN  !i.e. high cdr priority
C
            ICOSTINC=3
C
      ELSEIF(REASON(1:8).EQ.'TEMPORAL') THEN
C
            ICOSTINC=2
C
      ELSEIF(FAILPR(ITEM).GE.99) THEN
C
            ICOSTINC=1
      ELSE  !we blew it somewhere so scream
            WRITE(5,100) REASON(1:8),FAILPR(ITEM)
100         FORMAT(' Cost function blewup REASON >',A8,'< FAILPR=',I3)
            STOP
      ENDIF
C
C and then add it to the total....
C
      ICOST=ICOST+ICOSTINC
C
      ENDDO
      RETURN
      END
C==================================================================
      SUBROUTINE FAILCOPY(FAIL,FAIL_S,FAILPR,FAILPR_S)
C==================================================================
C  copies schedule fail list into save array
      CHARACTER FAIL*800,FAIL_S*800
```

```
      INTEGER FAILPR(41),FAILPR_S(41)
      DO I=1,41
            FAILPR_S(I)=FAILPR(I)
      ENDDO
      FAIL_S(1:800)=FAIL(1:800)
      RETURN
      END !hey, that was easy
C======================================================================
      SUBROUTINE HIGHPR(FAILPR,TRIED,IPNT)
C======================================================================
C  returns pointer to the highest cdr priority on the fail list
C  or zero if all have been tried
C
C  Only dichotomous vlues used in this current implementation
C
      INTEGER FAILPR(41),TRIED(40)
C
      IEND=FAILPR(41)-1  ! how many on the list now
      IVAL=99
      IPNT=0  ! in case we can't find one that has not been tried
      DO I=1,IEND
      IF((FAILPR(I) .LE. IVAL) .AND. (TRIED(I) .EQ. 0)) THEN
            IVAL=FAILPR(I)
            IPNT=I
      ENDIF
      ENDDO
      RETURN
      END
C======================================================================
      SUBROUTINE INIT
C======================================================================
C
C copies initial schedules etc into working arrays, zeros and
C  initializes other arrays and variables
C
      INCLUDE 'COMMON.FOR/LIST'
C
C copies schedules and and cdr priority vectors
C
      CALL SCHCOPY(SCHA_I,SCHA,SCHPRA_I,SCHPRA)
      CALL SCHCOPY(SCHB_I,SCHB,SCHPRB_I,SCHPRB)
      CALL SCHCOPY(SCHC_I,SCHC,SCHPRC_I,SCHPRC)
      CALL SCHCOPY(SCHD_I,SCHD,SCHPRD_I,SCHPRD)
      CALL SCHCOPY(SCHH_I,SCHH,SCHPRH_I,SCHPRH)
C
C zeros locked arrays
C
      DO I=1,40
            LOCKEDA(I)=0
            LOCKEDB(I)=0
            LOCKEDC(I)=0
            LOCKEDD(I)=0
            LOCKEDH(I)=0
      ENDDO
C
C initializes the pointer into the fail list
C
      FAILPRA(41)=1
      FAILPRB(41)=1
      FAILPRC(41)=1
      FAILPRD(41)=1
```

```
      FAILPRH(41)=1
C
      RETURN
      END
C==================================================================
      SUBROUTINE LCKSETUP (TEMP,L)
C==================================================================
C  sets up calendar with locked status of hours
C
      CHARACTER TEMP*480
      INTEGER L(40)
c     i=480
c     temp(i:i+20)='just for testing'
      DO I=1,40
      IPNT=(I-1)*12+1
      IF (L(I) .EQ. 0) THEN
              TEMP(IPNT:IPNT+11)='not locked   '
      ELSEIF (L(I).EQ.1) THEN
              TEMP(IPNT:IPNT+11)='SRC          '
      ELSEIF (L(I).EQ.2) THEN
              TEMP (IPNT:IPNT+11)='RESOURCES   '
      ELSEIF (L(I).EQ.3) THEN
              TEMP(IPNT:IPNT+11)='CDR PRIORITY'
      ELSEIF (L(I).EQ.4) THEN
              TEMP(IPNT:IPNT+11)='TEMPORAL    '
      ELSE
              WRITE(5,10) L(I),I
10            FORMAT(' bombed in LOCKSETUP.  LOCKED(I) & I =',2I8)
      ENDIF
      ENDDO
      RETURN
      END
C==================================================================
      SUBROUTINE PAFTER (TSK1,LOC,TSK2,SCH,LOCKED)
C==================================================================
C  This one attempts to resolve AFTER type precedence requirements.
C  If it fails it removes TSK1 from the schedule and puts it on the
C  FAIL list.  Only used after initial schedule are built.    6may85
      DIMENSION LOCKED(40)
      CHARACTER  TSK1*12,TSK2*12,SCH*480,TSKTMP*12
      INTEGER SCHPR(40)
C     TSKTMP(1:12)='            '
C  Check if TSK1 at hour one, if so move if anyplace
      IF (LOC .EQ. 1) THEN
1        ITO=RN(39)+1  !pick any hour between 2 and 40
         IF (LOCKED(ITO) .EQ. 1) GOTO 1  !if locked then get another,
c                                        !possible infinite loop*****
         CALL TSKSWAP(ITO,LOC,SCH,SCHPR)  !move it
         LOC=ITO  !keep track of where it went
         LOCKED(LOC)=1  !lock it
      ENDIF
C Is TSK2, the prerequisite, already in the schedule?
      CALL TSKSRCH(TSK2,SCH,1,IWHERE)
      IF (IWHERE .GT. 0) THEN        !Yes, in schedule
           IF (IWHERE .LT. LOC) RETURN !If already before then quit
15         ITO=RN(LOC-1)  !Select any hour before TSK1
           IF (LOCKED(ITO).EQ.1) GOTO 15  !if locked, get another
C                                          !possible infinite loop****
           CALL TSKSWAP(ITO,IWHERE,SCH,SCHPR)  !Interchange the
           LOCKED(ITO)=1      !lock it    !tasks arbitrarily
           RETURN  ! All done
```

```
      ELSE     !Prerequisite is not in table put will be put in
17          ITO=RN(LOC-1)  ! Select any hour before TSK1
            IF (LOCKED(ITO) .EQ. 1) GOTO 17 !if locked get another
            CALL TSKRPLC(ITO,TSK2,SCH)  !stuff it in
            LOCKED(ITO)=1  !lock it
      ENDIF
      RETURN
      END
C=================================================================
      subroutine prcd (sch)
C=================================================================
C
C This is used only during the building of schedules
C Checks temporal relationships for a given schedule
C  and attempts to resolve them  6MAY85 djg
C
      CHARACTER  SCH*480,TSK1*12,TSK2*480,REL*6
      CHARACTER  PRC*3000  !Precedence data, max of 100
      DIMENSION LOCKED(40)  !table of fixed tasks in terms
C                           of precedence resolution.
C
C If first time called read precedence table into memory
C
      IF (IFLAG .NE. 1) THEN
        IFLAG = 1 ! This is the first time so set flag
        OPEN (UNIT=3,FILE='PRCDTBL.DAT',STATUS='OLD')
        DO I=1,3000,30
          READ (3,10) PRC(I:I+29)
10        FORMAT(/A30)
        ENDDO
        CLOSE (UNIT=3)
      ENDIF
C
C Check every task in the schedule for precedence relationship
C
C     Clear the locked table for each schedule
        DO I=1,40
          LOCKED(I)=0
        ENDDO
      DO IHR = 1,40
      IPNT = (IHR-1)*12+1
      TSK1(1:12)=SCH(IPNT:IPNT+11)  !Move the task to dummy var
C Read up to 100 relationships
      DO IPRC = 1,100
        I=(IPRC-1)*30+1  !Pointer into precedence table
        IF(PRC(I:I+2) .EQ. 'END') GOTO 50  !End of table
        IF( TSK1(1:12) .EQ. PRC(I:I+11) )  THEN  !Found one
          REL(1:6)=PRC(I+12:I+17)  !Get the type relationship
          TSK2(1:12)=PRC(I+18:I+29) !Get the other task in relationship
            IF(REL(1:6) .EQ. 'AFTER ') THEN     !AFTER relationship
              CALL PAFTER(TSK1,IHR,TSK2,SCH,LOCKED)
              GOTO 30
            ENDIF
            WRITE(5,25) REL(1:6)  !This is an error state   Scream
25          FORMAT(' UNKNOWN RELATIONSHIP DETECTED:'/' I----I',
     $             /1X,A6)
30          CONTINUE  !jmp from precedence accommodation
        ENDIF
      ENDDO
50    CONTINUE  !JMP from end of data in precedence table
      ENDDO
```

```
      RETURN
      END
      SUBROUTINE PRISETUP(TEMP,PR)
C sets up commander priorities to be printed out
c
      CHARACTER TEMP*480
      INTEGER PR(40)
      DO I=1,40
      IPNT=(I-1)*12+1
      IF (PR(I).EQ.1) THEN
              TEMP(IPNT:IPNT+11)='PRIORITY    '
      ELSE
              TEMP(IPNT:IPNT+11)='            '
      ENDIF
      ENDDO
      RETURN
      END
C======================================================================
      SUBROUTINE PRTDSK (TITLE,SCH)
C======================================================================
C Writes a schedule to a disk file
      CHARACTER TITLE*40,SCH*480,A*12,B*12,C*12,D*12,E*12
      LU=2
      IFLAG=1
1     CONTINUE ! Jump to here if hardcopy requested
      WRITE(LU,5) TITLE(1:40)
5     FORMAT(//25X,A40)
      WRITE(LU,20)
20    FORMAT(/12X,'Monday',7X,'Tuesday',6X,'Wednesday',6X,
     * 'Thursday',7X,'Friday')
      WRITE(LU,25)
25    FORMAT(' TIME'/)
      I=1
      DO IHOUR=8,11
      A(1:12)=SCH(I:I+11)
      B(1:12)=SCH(I+96:I+107)
      C(1:12)=SCH(I+192:I+203)
      D(1:12)=SCH(I+288:I+299)
      E(1:12)=SCH(I+384:I+395)
      I=I+12
      WRITE(LU,30) IHOUR,A,B,C,D,E
30    FORMAT(/1X,I2,':','00',1X,5(2X,A12))
      ENDDO
      DO IHOUR=13,16
      A(1:12)=SCH(I:I+11)
      B(1:12)=SCH(I+96:I+107)
      C(1:12)=SCH(I+192:I+203)
      D(1:12)=SCH(I+288:I+299)
      E(1:12)=SCH(I+384:I+395)
      I=I+12
      WRITE(LU,30) IHOUR,A,B,C,D,E
      ENDDO
      RETURN
      END
C======================================================================
      SUBROUTINE PRTFAIL (FAIL, FAILPR)
C======================================================================
C
      CHARACTER FAIL*800,IN*1
      INTEGER FAILPR(41)
C
```

```
C DISPLAY THE FAIL LIST???
        write(5,140)
        READ(5,150) IN
140     FORMAT('$Do you want to view the fail list?')
150     format(a1)
        if( .not.((in(1:1).eq.'y').or.(in(1:1).eq.'Y'))) goto 200
        DO J=1,FAILPR(41)-1
        I=(J-1)*20+1
        WRITE(5,100) J,FAIL(I:I+19),FAILPR(J)
100     FORMAT(/1X,I4,4X,A20,I8)
        ENDDO
200     continue  !bypass print faillist
        RETURN
        END
C=====================================================================
        SUBROUTINE PRTFAIL2 (FAIL, FAILPR)
C=====================================================================
C Writes fail list to the disk
C
        CHARACTER FAIL*800,IN*1
        INTEGER FAILPR(41)
C
C writes the fail list to disk
        DO J=1,FAILPR(41)-1
        I=(J-1)*20+1
        WRITE(8,100) J,FAIL(I:I+19),FAILPR(J)
100     FORMAT(/1X,I4,4X,A20,I8)
        ENDDO
200     continue  !bypass print faillist
        RETURN
        END
C=====================================================================
        SUBROUTINE PRTSCH (TITLE,SCH)
C=====================================================================
C Prints a schedule to the screen or printer
        CHARACTER TITLE*40,SCH*480,A*12,B*12,C*12,D*12,E*12
C       CALL CLRSCRN
        LU=5
        IFLAG=1
1       CONTINUE ! Jump to here if hardcopy requested
        WRITE(LU,5) TITLE(1:40)
5       FORMAT(//25X,A40)
        WRITE(LU,20)
20      FORMAT(/12X,'Monday',7X,'Tuesday',6X,'Wednesday',6X,
     *   'Thursday',7X,'Friday')
        WRITE(LU,25)
25      FORMAT(' TIME'/)
        I=1
        DO IHOUR=8,11
        A(1:12)=SCH(I:I+11)
        B(1:12)=SCH(I+96:I+107)
        C(1:12)=SCH(I+192:I+203)
        D(1:12)=SCH(I+288:I+299)
        E(1:12)=SCH(I+384:I+395)
        I=I+12
        WRITE(LU,30) IHOUR,A,B,C,D,E
30      FORMAT(/1X,I2,':','00',1X,5(2X,A12))
        ENDDO
        DO IHOUR=13,16
        A(1:12)=SCH(I:I+11)
        B(1:12)=SCH(I+96:I+107)
```

```
        C(1:12)=SCH(I+192:I+203)
        D(1:12)=SCH(I+288:I+299)
        E(1:12)=SCH(I+384:I+395)
        I=I+12
        WRITE(LU,30) IHOUR,A,B,C,D,E
        ENDDO
C Prints hardcopy at user request
        IF( IFLAG .EQ. 1) THEN
          IFLAG=0
          LU=6
          WRITE(5,40)
40        FORMAT(/'$Do you want hardcopy?')
          READ (5,45) J
45        FORMAT(A1)
          IF( (J .EQ. 'Y') .OR. (J .EQ. 'y'))  GOTO 1
        ENDIF
        RETURN
        END
C=====================================================================
        SUBROUTINE PRTSCH2 (TITLE,SCH)
C=====================================================================
C Writes a schedule to a disk file
        CHARACTER TITLE*40,SCH*480,A*12,B*12,C*12,D*12,E*12
        WRITE(8,5) TITLE(1:40)
5       FORMAT(//25X,A40)
        WRITE(8,20)
20      FORMAT(/12X,'Monday',7X,'Tuesday',6X,'Wednesday',6X,
     *   'Thursday',7X,'Friday')
        WRITE(8,25)
25      FORMAT(' TIME'/)
        I=1
        DO IHOUR=8,11
        A(1:12)=SCH(I:I+11)
        B(1:12)=SCH(I+96:I+107)
        C(1:12)=SCH(I+192:I+203)
        D(1:12)=SCH(I+288:I+299)
        E(1:12)=SCH(I+384:I+395)
        I=I+12
        WRITE(8,30) IHOUR,A,B,C,D,E
30      FORMAT(/1X,I2,':','00',1X,5(2X,A12))
        ENDDO
        DO IHOUR=13,16
        A(1:12)=SCH(I:I+11)
        B(1:12)=SCH(I+96:I+107)
        C(1:12)=SCH(I+192:I+203)
        D(1:12)=SCH(I+288:I+299)
        E(1:12)=SCH(I+384:I+395)
        I=I+12
        WRITE(8,30) IHOUR,A,B,C,D,E
        ENDDO
        RETURN
        END
C=====================================================================
        SUBROUTINE REASON(LOCKED,IHR,REA)
C=====================================================================
C  gives the level for activities in the schedule being locked
C
        CHARACTER  REA*8
        INTEGER LOCKED(40)
C
        IF(LOCKED(IHR) EQ 0) THEN
```

```
              REA(1:8)='TIME    '
        ELSEIF (LOCKED(IHR).EQ.1) THEN
              REA(1:8)='SRC     '
        ELSEIF (LOCKED(IHR).EQ.2) THEN
              REA(1:8)='RESOURCE'
        ELSEIF (LOCKED(IHR).EQ.3) THEN
              REA(1:8)='CDR PRI '
        ELSEIF (LOCKED(IHR).EQ.4) THEN
              REA(1:8)='TEMPORAL'
        ENDIF
        RETURN
        END
C=====================================================================
        FUNCTION RN(N)
C=====================================================================
C Returns a random integer between 1 and N
        IF (ITEST .NE. 1) THEN  !NEEDED INITIALIZED RND WHEN
           ITEST = 1            !CALLED THE FIRST TIME!!!!!!
           Z=RND(-1)            !    ***SAVE***
        ENDIF
        IF(N.EQ.0) THEN !returns 0 if N is zero
           RN=0
           RETURN
        ENDIF
        RN= INT ( RND(0)*N)+1
        RETURN
        END
C=====================================================================
      FUNCTION RND(IFLAG)
C=====================================================================
C
C This little baby borrowed from A. Griesemer.   25Apr85 djg
c It IS VAX specific; uses the real time clock for seed
C
C     Produce psudo-random numbers in the range 0.0 to 1.0.
C     ISEED must be be initialized to a number between 1 and
C     2796202 the first time RND is called.  The argument IFLAG controls
C     this initialization.  If IFLAG is negitive the system clock is used
C     to initialize the seed.  If IFLAG is greater then zero the value of IFLAG
C     becomes the seed.  If IFLAG = 0 the previous ISEED is used to generate
C     the next random number in sequence.
C
C     Adapted form ACM algorithm 266[G5).
C
      IF(IFLAG) 10,40,30
10    ISEED=SECNDS(0.0)+1   !Reads the real time clock here.   djg
C      WRITE(3,15) ISEED
C15    FORMAT(' NOTE:  New value of random number seed is ',I6)
      GO TO 40
30    ISEED=IFLAG
C
40    ISEED=125*ISEED
      ISEED=ISEED-(ISEED/2796203)*2796203
      RND=FLOAT(ISEED)/2796203.0
      RETURN
      END
C=====================================================================
        SUBROUTINE RSRC
C=====================================================================
C  verifies the availability of resources for tasks
C
```

```
      CHARACTER TSKRS*1000,RSLST*400,IN*1,RSLSTS*400
      INTEGER RSTIME(50,40),RSQTY(50), RSTIMES(50,40),RSQTYS(50)
      CHARACTER TSK*12,INFILE*12
C
C
      INCLUDE 'COMMON.FOR/LIST'
C
      WRITE(5,1)
1     FORMAT(/' ***** Resolving Resource Constraints & Conflicts *****')
C
C  read the table of resources required by tasks
C  start off with fresh resource list each run
C
      IF (IFLAG.NE.1) THEN
      IFLAG=1
      OPEN (UNIT=3,FILE='TSKRSRC.DAT',STATUS='OLD')
      DO I=1, 50
         J=(I-1)*20+1   !compute pointer c1-12=task c13-20=resource
         READ(3,10)    TSKRS(J:J+19)
10       FORMAT(/A20)
      ENDDO
      CLOSE (UNIT=3)
C


C
C  read resource availability table
C
      OPEN (UNIT=3,FILE='RSRC.DAT',STATUS='OLD')
C
      DO I=1,50
        J=(I-1)*8+1   !pointer for resources
        READ(3,40) RSLSTS(J:J+7),RSQTYS(I),(RSTIMES(I,K),K=1,40)
40      FORMAT(/A8,I2,40I1)
      ENDDO
      CLOSE(UNIT=3)
      ENDIF
C read from saved values
      RSLST(1:400)=RSLSTS(1:400)
      DO I=1,50
      RSQTY(I)=RSQTYS(I)
      DO J=1,40
      RSTIME(I,J)=RSTIMES(I,J)
      ENDDO
      ENDDO
C
C  call the auxilliary routine for each company
C  NOTE:  The ordering of companies here gives implicit
C  prioritization to them.  First come, first served.
C
C     WRITE(5,139) FAILPRA(41)
C139  FORMAT('FAILPRA(41) on entry to RSRCAUX =',i4)
      CALL RSRCAUX(SCHA,SCHPRA,FAILA,FAILPRA,LOCKEDA,
     $       TSKRS,RSLST,RSTIME,RSQTY)
C
      CALL RSRCAUX(SCHB,SCHPRB,FAILB,FAILPRB,LOCKEDB,
     $       TSKRS,RSLST,RSTIME,RSQTY)
C
      CALL RSRCAUX(SCHC,SCHPRC,FAILC,FAILPRC,LOCKEDC,
     $       TSKRS,RSLST,RSTIME,RSQTY)
C
      CALL RSRCAUX(SCHD,SCHPRD,FAILD,FAILPRD,LOCKEDD,
```

```
     $        TSKRS, RSLST, RSTIME, RSQTY)
C
           CALL RSRCAUX(SCHH, SCHPRH, FAILH, FAILPRH, LOCKEDH,
     $        TSKRS, RSLST, RSTIME, RSQTY)
C
           RETURN
           END
C=====================================================================
           SUBROUTINE RSRCAUX(SCH, SCHPR, FAIL, FAILPR, LOCKED,
     $      TSKRS, RSLST, RSTIME, RSQTY)
C=====================================================================
C
C   does the resource work on each company
C   this version accommodates but one renewable and
C    one expendable resource per activity
C
           CHARACTER TSKRS*1000, RSLST*400
           INTEGER RSTIME(50, 40), RSQTY(50), ITIME(40)
C
           CHARACTER SCH*480, FAIL*800, TSK*12, RS*8
           INTEGER SCHPR(40), LOCKED(40), FAILPR(41)
C
           DO IHR=1, 40  !for every task on the schedule
            IF (LOCKED(IHR).EQ.2) GOTO 100 !previously resource locked
            ITSK=(IHR-1)*12+1  !pointer to the task
            TSK(1:12)=SCH(ITSK:ITSK+11)
            DO IRSC=1, 50  !check the resources need table
               ITSKRS=(IRSC-1)*20+1   !pointer into the resource req table
               IF (TSKRS(ITSKRS:ITSKRS+2) .EQ  'END') GOTO 100  !end of table
               IF (TSK(1:12).EQ.TSKRS(ITSK RS:ITSKRS+11)) THEN  !needs resources
                  RS(1:8)=TSKRS(ITSKRS+12:ITSKRS+19)  !get the resource id
                  DO IAVAIL=1, 50  ! check the resource available table
                     IAVPNT=(IAVAIL-1)*8+1
                     IF (RSLST(IAVPNT:IAVPNT+2).EQ.'END') GOTO 90
                     IF (RS(1:8).EQ.RSLST(IAVPNT:IAVPNT+7)) THEN !rs available
                         IF(RSQTY(IAVAIL).GT.0) THEN  !qty available
                            RSQTY(IAVAIL)=RSQTY(IAVAIL)-1 !decr qty
           IF (LOCKED(IHR).EQ.0)LOCKED(IHR)=2  !it's go.  lock if not already
                         ELSEIF(RSTIME(IAVAIL, IHR).GT.0) THEN  !renewable avail
                            RSTIME(IAVAIL, IHR)=RSTIME(IAVAIL, IHR)-1 !decr
           IF (LOCKED(IHR).EQ.0)LOCKED(IHR)=2  !lock it, if not already
                  ELSE
C   here check if renewable resource is available at another time
C   and if so we swap it with a nonlocked task at that time
                            DO I=1, 40 !copy into working array
                               ITIME(I)=RSTIME(IAVAIL, I)
                            ENDDO
45                          ISUM=0
                            DO I=1, 40
                               ISUM=ISUM+ITIME(I)
                            ENDDO
C   if very many times thru here then just give up
                            IF(ISUM.GT.0) THEN  !resource avail sometime
47                             IHRNEW=RN(40)  !get a random time
                               IF (ITIME(IHRNEW).EQ.0) GOTO 47!no good
                               IF (LOCKED(IHRNEW).GT.0) THEN  !no good
                                  ITIME(IHRNEW)=0
                                  GOTO 45
                               ENDIF
C make sure activity at target doesn't require any resources  3june djn
           DO I=1, 50
```

```
                IPNT2=(I-1)*20+1 !pointer into RSC
                IPNT3=(IHRNEW-1)*12+1 !pointer into target
                IF(SCH(IPNT3:IPNT3+11).EQ.TSKRS(IPNT2:IPNT2+11)) THEN
                        ITIME(IHRNEW)=0
                        GOTO 45
                ENDIF
        ENDDO
                                CALL TSKSWAP(IHRNEW,IHR,SCH,SCHPR)
                                LOCKED(IHRNEW)=2
                                RSTIME(IAVAIL,IHRNEW)=RSTIME(IAVAIL,IHRNEW)-1
                                GOTO 49 !success, jump out
                            ENDIF
C  if drop through to this point then record fail condition
99                          CONTINUE  !failure
                            IFLPNT=(FAILPR(41)-1)*20+1
                            FAIL(IFLPNT:IFLPNT+11)=TSK(1:12)   !task
C  if failure of SRC item then record same
                            IF (LOCKED(IHR).EQ.1) THEN
                                LOCKED(IHR)=0
                                FAIL(IFLPNT+12:IFLPNT+19)='SRC      '
                            ELSE
                                FAIL(IFLPNT+12:IFLPNT+19)='RESOURCE' !reason
                            ENDIF
                            FAILPR(FAILPR(41))=SCHPR(IHR) !save priority
                            FAILPR(41)=FAILPR(41)+1   !inc the pointer
                            SCH(ITSK,ITSK+11)='            '
                            SCHPR(IHR)=99
49                          CONTINUE  ! jump here if successful time swap
                        ENDIF
                    ENDIF
                ENDDO
90                  CONTINUE  !end of resource availability table
              ENDIF
          ENDDO
100         CONTINUE  !end of resources needed table
        ENDDO
        RETURN
        END
C======================================================================
        SUBROUTINE SAVE(ICOST,ICOSTMIN)
C======================================================================
C
C saves results of iteration if first or best so far
C
        INTEGER DUMMY(40)

        INCLUDE 'COMMON.FOR/LIST'
C
C       WRITE(*,*) ICOST
C
        IF (IFIRST.NE.1) THEN  !if very first call then initialize
                IFIRST=1
                ICOSTMIN=1000000  !nice and big; play it safe
        ENDIF
C
C write the cost to file for004.dat
C
        WRITE(4,50) ICOST
50      FORMAT(I8)
C
C if not best so far then forget it
```

```
C
        IF(ICOST.GE.ICOSTMIN) THEN
        RETURN
        ENDIF
C
C otherwise, save the world
C
        ICOSTMIN=ICOST
C
        CALL SCHCOPY(SCHA,SCHA_S,DUMMY,DUMMY)
        CALL SCHCOPY(SCHB,SCHB_S,DUMMY,DUMMY)
        CALL SCHCOPY(SCHC,SCHC_S,DUMMY,DUMMY)
        CALL SCHCOPY(SCHD,SCHD_S,DUMMY,DUMMY)
        CALL SCHCOPY(SCHH,SCHH_S,DUMMY,DUMMY)
C
        CALL FAILCOPY(FAILA,FAILA_S,FAILPRA,FAILPRA_S)
        CALL FAILCOPY(FAILB,FAILB_S,FAILPRB,FAILPRB_S)
        CALL FAILCOPY(FAILC,FAILC_S,FAILPRC,FAILPRC_S)
        CALL FAILCOPY(FAILD,FAILD_S,FAILPRD,FAILPRD_S)
        CALL FAILCOPY(FAILH,FAILH_S,FAILPRH,FAILPRH_S)
C
        RETURN
        END
C=====================================================================
        SUBROUTINE SAVEALL(HEADING,ITER)
C=====================================================================
C
C  saves the world to disk
C
        CHARACTER HEADING*40,TEMP*480
C
        INCLUDE 'COMMON.FOR/LIST'
C
C  write to file FOR002.DAT
C
        WRITE(2,5) HEADING,ITER
5       FORMAT(/1X,A40,'     ITERATION = ',I6)
C
C  company A first
        CALL PRTDSK(A,SCHA)   !same as SCHPRT but LU=2
C
        CALL PRISETUP(TEMP,SCHPRA)
C
        CALL PRTDSK(A,TEMP)
C
        CALL LCKSETUP(TEMP,LOCKEDA)
C
        CALL PRTDSK(A,TEMP)
C
C  print fail list and priority
C
        DO I=1,FAILPRA(41)-1
                IPNT=(I-1)*20+1
                WRITE(2,10)  FAILA(IPNT:IPNT+19),FAILPRA(I)
10              FORMAT (1X,A20,5X,I3)
        ENDDO
C  company B next
        CALL PRTDSK(B,SCHB)   !same as SCHPRT but LU=2
C
        CALL PRISETUP(TEMP,SCHPRB)
C
```

```
      CALL PRTDSK(B,TEMP)
C
      CALL LCKSETUP(TEMP,LOCKEDB)
C
      CALL PRTDSK(B,TEMP)
C
C  print fail list and priority
C
      DO I=1,FAILPRB(41)-1
            IPNT=(I-1)*20+1
            WRITE(2,10)  FAILB(IPNT:IPNT+19),FAILPRB(I)
      ENDDO
C  company C next
      CALL PRTDSK(C,SCHC)  !same as SCHPRT but LU=2
C
      CALL PRISETUP(TEMP,SCHPRC)
C
      CALL PRTDSK(C,TEMP)
C
      CALL LCKSETUP(TEMP,LOCKEDC)
C
      CALL PRTDSK(C,TEMP)
C
C  print fail list and priority
C
      DO I=1,FAILPRC(41)-1
            IPNT=(I-1)*20+1
            WRITE(2,10)  FAILC(IPNT:IPNT+19),FAILPRC(I)
      ENDDO
C  company D next
      CALL PRTDSK(D,SCHD)  !same as SCHPRT but LU=2
C
      CALL PRISETUP(TEMP,SCHPRD)
C
      CALL PRTDSK(D,TEMP)
C
      CALL LCKSETUP(TEMP,LOCKEDD)
C
      CALL PRTDSK(D,TEMP)
C
C  print fail list and priority
C
      DO I=1,FAILPRD(41)-1
            IPNT=(I-1)*20+1
            WRITE(2,10)  FAILD(IPNT:IPNT+19),FAILPRD(I)
      ENDDO
C  company H next
      CALL PRTDSK(HHC,SCHH)  !same as SCHPRT but LU=2
C
      CALL PRISETUP(TEMP,SCHPRH)
C
      CALL PRTDSK(HHC,TEMP)
C
      CALL LCKSETUP(TEMP,LOCKEDH)
C
      CALL PRTDSK(HHC,TEMP)
C
C  print fail list and priority
C
      DO I=1,FAILPRH(41)-1
            IPNT=(I-1)*20+1
```

```
              WRITE(2,10)  FAILH(IPNT:IPNT+19),FAILPRH(I)
        ENDDO

        RETURN
        END
C=================================================================
        SUBROUTINE SCHBLD
C=================================================================
C Builds random schedules for each of five companies using TSKLST.
C
        INCLUDE 'COMMON.FOR/LIST'
C
        WRITE(5,10)
10      FORMAT(/' ***** Building Company Training Schedules *****')
C
        CALL SCHBLDAX(SCHA_I,SCHPRA_I)
        CALL SCHBLDAX(SCHB_I,SCHPRB_I)
        CALL SCHBLDAX(SCHC_I,SCHPRC_I)
        CALL SCHBLDAX(SCHD_I,SCHPRD_I)
        CALL SCHBLDAX(SCHH_I,SCHPRH_I)

C
        RETURN
        END
C=================================================================
        SUBROUTINE SCHBLDAX (SCH,SCHPR)
C=================================================================
C Builds random schedules for each of five companies using TSKLST.DAT
        CHARACTER SCHTSKS*1200,SCH*480,IN*1
        INTEGER SCHPR(40),USED(100)
C
C  reads previously generated schedules
C
        IF (IFLAG4.EQ.1) GOTO 6
        IF (IFLAG3.EQ.1) GOTO 4   ! already reading old schedules
        WRITE(5,2)
2       FORMAT('$Do you want previously generated initial schedules?')
        READ(5,3) IN
3       FORMAT(A1)
        IF( (IN(1:1).EQ.'Y') .OR. (IN(1:1).EQ.'y')) THEN
                IFLAG3=1
        OPEN (UNIT=7,FILE='SAVESCH.DAT',STATUS='OLD')
4               CONTINUE
                DO I=1,40
                IPT=(I-1)*12+1
                READ(7,5) SCH(IPT:IPT+11),SCHPR(I)
5               FORMAT(A12,I3)
                ENDDO
        RETURN
        ENDIF
        IFLAG4=1 ! no old files
6       CONTINUE
C
        DO I=1,100
                USED(I)=0
        ENDDO
C
C Read TSKLST first time called
        IF (IFLAG .NE. 1) THEN
           IFLAG = 1
           OPEN (UNIT = 3, FILE ='TSKLST.DAT', STATUS='OLD')
```

```
            DO I= 1,100
            USED(I)=0  !clear this guy at the same time
            J=(I-1)*12+1
            READ(3,10) SCHTSKS(J:J+11)
10          FORMAT(/4XA12)
            ENDDO
            CLOSE (UNIT= 3)
         ENDIF
C Randomly build training schedule
         DO IHOUR=1,40
15       CONTINUE
         KK=RN(60)
         IF(USED(KK).EQ.1) GOTO 15  !used this task so get another
         USED(KK)=1  ! mark it used
         K= (KK-1)*12+1
         J= (IHOUR-1)*12+1
         SCH(J:J+11) = SCHTSKS(K:K+11)
         ENDDO
C generate some cdr priorities for the tasks
         DO I=1,40
C           SCHPR(I)=RN(40)!this just gives random values 1 to 40
c this will give dichotomous values: 1=priorities (20%) 99=no pri
C
                SCHPR(I)=99
         ENDDO
         DO I=1,8
499             CONTINUE
                J=RN(40)
                IF(SCHPR(J).EQ.1) GOTO 499  !already a cdr priority
                SCHPR(J)=1
         ENDDO
C save the schedules
         IF(IFLAG2.EQ.1) GOTO 500 !file already open
         OPEN (UNIT=7,FILE='SAVESCH.DAT',STATUS='OLD')
500      CONTINUE
         IFLAG2=1
         DO I=1,40
         IPT=(I-1)*12+1
         WRITE (7,5) SCH(IPT:IPT+11),SCHPR(I)
         ENDDO
         RETURN
         END
C=====================================================================
         SUBROUTINE SCHCOPY(IN,OUT,SCHPRIN,SCHPROUT)
C=====================================================================
C copies one schedule into another
         CHARACTER IN*480,OUT*480
         INTEGER SCHPRIN(40),SCHPROUT(40)
         DO I=1,40
                SCHPROUT(I)=SCHPRIN(I)
         ENDDO
         OUT(1:480)=IN(1:480)
         RETURN
         END
C=====================================================================
         SUBROUTINE SRC
C=====================================================================
C Propagates the SHORT-RANGE CALENDAR downward
C
         CHARACTER SRCTSK*1200,SRCUNIT*100,TSK*12,INFILE*12,UNIT*1
         INTEGER HOUR(100)
```

```
C This common is only with SRCAUX
C
        INCLUDE 'COMMON.FOR/LIST'
C
        WRITE(5,1)
1       FORMAT(/' ***** Inheriting Short-range Calendar Activities *****')
C
                INFILE(1:12)='SRC.DAT'
C  Read the SRC.DAT file
c  first entry only!
        IF(IFLAG.NE.1) THEN
        IFLAG=1
c
        OPEN (UNIT=3,FILE=INFILE,STATUS='OLD')
C
        DO I=1,100    !MAX OF 100 ITEMS IN SRC
           K=(I-1)*12+1
           READ(3,20) SRCTSK(K:K+11),SRCUNIT(I:I),HOUR(I)
20         FORMAT(/A12,1X,1A1,1X,I2)
        ENDDO
        CLOSE (UNIT=3)
        ENDIF
C
C  Now do it for each of the five units
C
           FAILPRA(41)=1   !should be in INIT subroutine!!!!!
           UNIT='A'
        CALL SRCAUX(SCHA,SCHPRA,FAILA,UNIT,FAILPRA,LOCKEDA,SRCUNIT,
     $      SRCTSK,HOUR)
C
           UNIT='B'
        CALL SRCAUX(SCHB,SCHPRB,FAILB,UNIT,FAILPRB,LOCKEDB,SRCUNIT,
     $      SRCTSK,HOUR)
C
           UNIT='C'
        CALL SRCAUX(SCHC,SCHPRC,FAILC,UNIT,FAILPRC,LOCKEDC,SRCUNIT,
     $      SRCTSK,HOUR)
C
           UNIT='D'
        CALL SRCAUX(SCHD,SCHPRD,FAILD,UNIT,FAILPRD,LOCKEDD,SRCUNIT,
     $      SRCTSK,HOUR)
C
           UNIT='H'
        CALL SRCAUX(SCHH,SCHPRH,FAILH,UNIT,FAILPRH,LOCKEDH,SRCUNIT,
     $      SRCTSK,HOUR)
C
        RETURN
        END
C=====================================================================
        SUBROUTINE SRCAUX(SCH,SCHPR,FAIL,IUNIT,FAILPR,LOCKED,SRCUNIT,
     $       SRCTSK,HOUR)
C=====================================================================
C       This is the workhorse src   8MAY85  djg
        CHARACTER SCH*480,FAIL*800,TSK*12,SRCTSK*1200,SRCUNIT*100,
     $     IUNIT*1
        INTEGER HOUR(100),FAILPR(41),SCHPR(40),LOCKED(40)
C
C  Run down the SRC inserting activities in the schedule, putting dis-
C  placed items in FAIL with the cdr's priority in FAILPR
C
        DO ISRC =1,100   !max of 100 items read
```

```
          I=(ISRC-1)*12+1
        IF (SRCTSK(I:I+2).EQ.'END') GOTO 50  !quit if end of SRC
C  See if this activitiy is relevant to this unit
        IF ( (SRCUNIT(ISRC:ISRC).EQ.IUNIT) .OR.
     *       (SRCUNIT(ISRC:ISRC).EQ. 'E')) THEN
C  If hour unspecified, then get one that is not locked
                J=HOUR(ISRC)
                IF (J.EQ.O) THEN
10                      J=RN(40)
                        IF (LOCKED(J) .GE. 1) GOTO 10  !it's locked, get
                ENDIF                                  !another
C                                                      !posssilbe LOOP
                K= (J-1)*12+1  !pointer into schedule
                TSK(1:12)=SCH(K:K+11)  !save whatever is in there
                I= (FAILPR(41)-1)*20+1  !calc pointer into fail list
                FAIL(I:I+11)=TSK(1:12) !put on fail list
                I= I+12                 !pointer for reason
C               IF(LOCKED(J).GE.1) THEN !if an SRC item here then move it
C20                     IPNT=RN(40) !get an hour
C                       IF (LOCKED(IPNT).EQ.1) GOTO 20 !if locked get another
C                       CALL TSKSWAP(IPNT,J,SCH)          !POSSIBLE LOOP HERE
C                       LOCKED(IPNT)=1
C               ENDIF    !this block obviated by imposed structure on SRC.DAT
                FAIL(I:I+7)='TIME    '   !save the reason
                FAILPR(FAILPR(41))=SCHPR(J)  !save the priority
                FAILPR(41)=FAILPR(41)+1  !inc pointer into fail arrays
                I=(ISRC-1)*12+1  !pointer into src
                SCH(K:K+11)=SRCTSK(I:I+11)  !put the SRC tsk on SCH
                LOCKED(J)=1   !now lock the hour
                SCHPR(J)=99
        ENDIF
        ENDDO
50      CONTINUE  !jump from end of SRC
        RETURN
        END
C===================================================================
        SUBROUTINE TAFTER(SCH,SCHPR,FAIL,FAILPR,TSK1,IHR,TSK2,LOCKED)
C===================================================================
C this guy assures that TSK2 precedes TSK1 in the schedule or TSK1
c is added to the fail list.
c
        CHARACTER SCH*480,FAIL*800,TSK1*12,TSK2*12,REA*8
        INTEGER SCHPR(40),FAILPR(41),LOCKED(40),IDUM2(40)
C
C  is tsk2 already on the schedule before tsk1, if so quit
c
        IBLKFLG=0  !assume there are no unlocked times in schedule
        DO IHR2=1,IHR-1  !see if prereq. already preceeds TSK!
                IF(LOCKED(IHR2) .EQ. O) IBLKFLG=1  !found an unlocked time
                IPNT=(IHR2-1)*12+1
                IF (SCH(IPNT:IPNT+11) .EQ. TSK2)  RETURN  !yes, all done
        ENDDO
C  prerequisite is not on the schedule.  If unlocked hours then stuff
C  it into one of the hours.
        ICOUNT=0
        IF(IBLKFLG.EQ.1) THEN !got >=  1  unlocked one, so find it
10              I=RN(IHR-1) !pick any ol' one
                ICOUNT=ICOUNT+1     ! if here VERY long then give up
                IF(ICOUNT.GT.9999) GOTO 99  !cheap loop avoidance ..
                IF(SCH( (I-1)*12+1. (I-1)*12+12).EQ.'            '
     *           ) GOTO 10 !no blanks please
```

```
                IF(LOCKED(I).GE. 1) GOTO 10  !no good,get another
                TSK1(1:12)=SCH( (I-1)*12+1:(I-1)*12+12)  !save it
                SCH ((I-1)*12+1:(I-1)*12+12)=TSK2(1:12)
                LOCKED(I)=4 !i.e. locked by temporal relationship
                J=(FAILPR(41)-1)*20+1  !pointer into fail list
                FAIL(J:J+11)=TSK1  !what was orignially there
                FAIL(J+12:J+19)='TIME    '
                FAILPR(FAILPR(41))=SCHPR(I)
                FAILPR(41)=FAILPR(41)+1  !inc the fail list pointer
                RETURN    !all done
        ENDIF
C
C  if here then TSK1 becomes the fail item and blanks go into the schedule
c
c       write(5,105) locked,ihr,rea
c105    format(' entering REASON',40i2,i6,a10)
99      CONTINUE  ! give up
        CALL REASON (LOCKED,IHR,REA)  !get the reason tsk was locked
C
        TSK2(1:12)='           '
c       write(5,110)
c110    format(' entering TSKSUB')
c       write(5,*) IHR,TSK2,TSK1,FAIL,FAILPR,IDUMMY,REA,IDUM2
        CALL TSKSUB(SCH,SCHPR,IHR,TSK2,TSK1,FAIL,FAILPR,IDUMMY,REA,IDUM2)
        RETURN
        END
C=====================================================================
        SUBROUTINE TIMMAFT(SCH,SCHPR,FAIL,FAILPR,TSK1,IHR,TSK2,LOCKED)
C=====================================================================
C  checks for relation that TSK1 must follow immediately after TSK2.  If
C  not poosible TSK1 is added to the unscheduled list
C
C
C
        CHARACTER SCH*480,FAIL*800,TSK1*12,TSK2*12,REA*8
        INTEGER SCHPR(40),FAILPR(41),LOCKED(40),IDUM2(40)
C
C if it already there then return
C
        IPNT=(IHR-1)*12+1  !pointer into schedule
        IPNT2=(IHR-2)*12+1 !pointer to the hour before
        IF (SCH(IPNT2:IPNT2+11).EQ.TSK2(1:12)) THEN
                RETURN
        ENDIF
C
C it's not always that easy.  check locked time
C
        IF(LOCKED(IHR-1).EQ.0) THEN  ! if unlocked then stuff it in
                I=IHR-1
                TSK1(1:12)=SCH( (I-1)*12+1:(I-1)*12+12)  !save it
                SCH ((I-1)*12+1:(I-1)*12+12)=TSK2(1:12)
                LOCKED(I)=4 !i.e. locked by temporal relationship
                J=(FAILPR(41)-1)*20+1  !pointer into fail list
                FAIL(J:J+11)=TSK1  !what was orignially there
                FAIL(J+12:J+19)='TIME    '
                FAILPR(FAILPR(41))=SCHPR(I)
                FAILPR(41)=FAILPR(41)+1  !inc the fail list pointer
                RETURN    !all done
        ENDIF
C
C  if here then TSK1 becomes the fail item and blanks go into the schedule
```

```
c
        CALL REASON (LOCKED,IHR,REA)  !get the reason tsk was locked
C
        TSK2(1:12)='            '
        LOCKED(IHR)=0
        CALL TSKSUB(SCH,SCHPR,IHR,TSK2,TSK1,FAIL,FAILPR,IDUMMY,REA,IDUM2)
        RETURN
        END
C==================================================================
        SUBROUTINE TIMMBEF(SCH,SCHPR,FAIL,FAILPR,TSK1,IHR,TSK2,LOCKED)
C==================================================================
C  checks for relation that TSK1 must be follow immediately by TSK2.  If
C  not poosible TSK1 is added to the unscheduled list


c
        CHARACTER SCH*480,FAIL*800,TSK1*12,TSK2*12,REA*8
        INTEGER SCHPR(40),FAILPR(41),LOCKED(40),IDUM2(40)
C

C
C if it already there then return
C
        IF(IHR.EQ.40) GOTO 99  !impossible to satisfy
        IPNT=(IHR-1)*12+1  !pointer into schedule
        IPNT2=(IHR-0)*12+1 !pointer to the hour after
        IF (SCH(IPNT2:IPNT2+11).EQ.TSK2(1:12)) THEN
                IF(LOCKED(IHR).EQ.0) LOCKED(IHR)=4
                IF(LOCKED(IHR+1).EQ.0) LOCKED(IHR+1)=4
                RETURN
        ENDIF
C
C it's not always that easy.  check locked time
C
        IF(LOCKED(IHR+1).EQ.0) THEN  ! if unlocked then stuff it in
                I=IHR+1
                TSK1(1:12)=SCH( (I-1)*12+1:(I-1)*12+12)  !save it
                SCH ((I-1)*12+1:(I-1)*12+12)=TSK2(1:12)
                LOCKED(I)=4 !i.e. locked by temporal relationship
                J=(FAILPR(41)-1)*20+1  !pointer to fail list
C               WRITE(5,501) FAILPR(41),J
C501    FORMAT(' FAILPR(41) & J= ', 2I8)
                FAIL(J:J+11)=TSK1(1:12)  !what was orignially there
                FAIL(J+12:J+19)='TIME    '
                FAILPR(FAILPR(41))=SCHPR(I)
                FAILPR(41)=FAILPR(41)+1  !inc the fail list pointer
                RETURN    !all done
        ENDIF
C
C  if here then TSK1 becomes the fail item and blanks go into the schedule
c
99      CONTINUE  !fail condition
        CALL REASON (LOCKED,IHR,REA)  !get the reason tsk was locked
C
        TSK2(1:12)='            '
        CALL TSKSUB(SCH,SCHPR,IHR,TSK2,TSK1,FAIL,FAILPR,IDUMMY,REA,IDUM2)
        RETURN
        END
C==================================================================
        SUBROUTINE TIMMFOL(SCH,SCHPR,FAIL,FAILPR,TSK1,IHR,TSK2,LOCKED)
C==================================================================
```

```
C  checks for relation that TSK1 must follow immediately after TSK2.  If
C  not TSK1 is added to the unscheduled list. NO ATTEMPT IS MADE TO
C  INSERT TSK2!  (I.E. TASKS ARE ASSUMED TO BE PART OF A BLOCK)
C
C
C
        CHARACTER SCH*480,FAIL*800,TSK1*12,TSK2*12,REA*8
        INTEGER SCHPR(40),FAILPR(41),LOCKED(40),IDUM2(40)
C
C if it already there then return
C
        IPNT=(IHR-1)*12+1   !pointer into schedule
        IPNT2=(IHR-2)*12+1  !pointer to the hour before
        IF (SCH(IPNT2:IPNT2+11).EQ.TSK2(1:12)) THEN
                RETURN
        ENDIF
C
C
C  if here then TSK1 becomes the fail item and blanks go into the schedule
c
C get the reason this activity was previosly locked, if any
c

C       IF (LOCKED(IHR).NE.0) THEN
                CALL REASON(LOCKED,IHR,REA)
C       ELSE
C               REA(1:8)='TEMPORAL'
C       ENDIF
C
        TSK2(1:12)='            '
        LOCKED(IHR)=0
        CALL TSKSUB(SCH,SCHPR,IHR,TSK2,TSK1,FAIL,FAILPR,IDUMMY,REA,IDUM2)
        RETURN
        END
C==================================================================
        subroutine TMPRL
C==================================================================
C
C Checks all precedence relationships for each company
C This implementation assumes that an activity on the
C  right side in the temporal relationship table requires
C  no resources.
C
C       CHARACTER  SCH*480,TSK1*12,TSK2*480,REL*6
        CHARACTER  PRC*3000  !Precedence data, max of 100
                             !table of fixed tasks in terms
C                             of precedence resolution.
C read common block
c
        INCLUDE 'COMMON.FOR/LIST'
c
        WRITE(5,3)
3       FORMAT(/' ***** Verifying and Resolving Temporal Relationships *****')
C
C Read precedence table into memory
C
C  only need to read once
C
        IF(IRDFLAG.NE.1) THEN
        IRDFLAG=1
           OPEN (UNIT=3,FILE='PRCDTBL.DAT',STATUS='OLD')
```

```
            DO I=1,3000,30
              READ (3,10) PRC(I:I+29)
10            FORMAT(/A30)
            ENDDO
            CLOSE (UNIT=3)
          ENDIF
C
C  Check it for each company
C
          CALL TMPRLAUX (SCHA,SCHPRA,FAILA,FAILPRA,LOCKEDA,PRC)
          CALL TMPRLAUX (SCHB,SCHPRB,FAILB,FAILPRB,LOCKEDB,PRC)
          CALL TMPRLAUX (SCHD,SCHPRD,FAILD,FAILPRD,LOCKEDD,PRC)
          CALL TMPRLAUX (SCHC,SCHPRC,FAILC,FAILPRC,LOCKEDC,PRC)
          CALL TMPRLAUX (SCHH,SCHPRH,FAILH,FAILPRH,LOCKEDH,PRC)
C
          RETURN
          END
C=======================================================================
          SUBROUTINE TMPRLAUX (SCH,SCHPR,FAIL,FAILPR,LOCKED,PRC)
C=======================================================================
C
C does the checking for temporal violations for each company
C
          CHARACTER SCH*480,FAIL*800,PRC*3000,TSK1*12,TSK2*12,REL*6
          INTEGER SCHPR(40),FAILPR(41),LOCKED(40)
C
C NEED TO DO REPEATEDLY UNTIL NO CHANGES ARE MADE
C  later maybe, now just get through it once
C
          IF(FAILPR(41).EQ.0) THEN
                WRITE(5,113)
113       FORMAT(' FAIL(PR) IS ZERO*****************')
          ENDIF
C
c             IFLAG=1  ! set the repeat flag for first time though
C         DO WHILE (IFLAG .EQ. 1)
C             IFLAG=0  ! but this may the last time through
C
          DO IHR=1,40     ! do for every task in the schedule
          ITSK=(IHR-1)*12+1   !pointer into sch
C do up to 100 relationships
             DO IPRC=1,100  ! row in temporal relationship table
             IPNT=(IPRC-1)*30+1 !pointer into temporal tbl array
             IF (PRC(IPNT:IPNT+2) .EQ. 'END') GOTO 1000  !end of data
             TSK1(1:12)=SCH(ITSK:ITSK+11)  !pull the tsk from the schedule
             IF(TSK1(1:12) .EQ. PRC(IPNT:IPNT+11)) THEN !got a match
C                IFLAG=1  !set the repeat flag for the while loop
C now let's see what kind of relationship we have
                 REL(1:6)=PRC(IPNT+12:IPNT+17)
                 TSK2(1:12)=PRC(IPNT+18:IPNT+29) !get the paired task too
                 IF (REL(1:6) .EQ. 'AFTER ') CALL TAFTER(SCH,SCHPR,FAIL,FAILPR,
     $                 TSK1,IHR,TSK2,LOCKED)
                 IF (REL(1:6) .EQ. 'IMMAFT') CALL TIMMAFT(SCH,SCHPR,FAIL,FAILPR,
     $                 TSK1,IHR,TSK2,LOCKED)
                 IF (REL(1:6) .EQ. 'IMMBEF') CALL TIMMBEF(SCH,SCHPR,FAIL,FAILPR,
     $                 TSK1,IHR,TSK2,LOCKED)
                 IF (REL(1:6) .EQ. 'IMMFOL') CALL TIMMFOL(SCH,SCHPR,FAIL,FAILPR,
     $                 TSK1,IHR,TSK2,LOCKED)
              ENDIF
              ENDDO
1000          CONTINUE  !end of data in PRC table
```

```
        ENDDO
C       ENDDO
        RETURN
        END
C=====================================================================
        SUBROUTINE TSKRPLC(LOC,TSK,SCH)
C=====================================================================
C Puts task into a schedule at location LOC.
        CHARACTER   SCH*480,TSK*12
        I=(LOC-1)*12+1
        SCH(I:I+11)=TSK(1:12)
        RETURN
        END
        SUBROUTINE TSKSRCH (TSK,SCH,ISTART,LOC)
C Searchs given schedule for a task from hour ISTART (1-40).  Returns
C hour location or zero if not found.
C
        CHARACTER TSK*12,SCH*480
        DO I = ISTART,40
          J=(I-1)*12+1
            IF (TSK(1:12) .EQ  SCH (J:J+11)) THEN
                LOC=I
                RETURN
            ENDIF
        ENDDO
        LOC=0
        RETURN
        END
C=====================================================================
        SUBROUTINE TSKSUB(SCH,SCHPR,ITARGET,TSKIN,TSKOUT,
     $     FAIL,FAILPR,IPNT,REASON,TRIED)
C=====================================================================
C substitutes TSKIN in schedule SCH at hour ITARGET, removing that task
C TSKOUT, putting it on the fail list with REASON.  Also swaps priorities.
C
        CHARACTER SCH*480,TSKIN*12,TSKOUT*12,FAIL*800,REASON*8
        INTEGER SCHPR(40),FAILPR(41),TRIED(40)
C       CHARACTER DUMMY*40
C
        ISCH=(ITARGET-1)*12+1
        TSKOUT(1:12)=SCH(ISCH:ISCH+11)
        SCH(ISCH:ISCH+11)=TSKIN(1:12)
        ISAVE=SCHPR(ITARGET)
        IF (TSKIN(1:12).EQ.'            ') THEN  !special case when
                SCHPR(ITARGET)=1000              !TSKIN is blank
                IPNT=FAILPR(41)
                FAILPR(41)=FAILPR(41)+1
        ELSE
                SCHPR(ITARGET)=FAILPR(IPNT)
        ENDIF
        IF (TSKOUT(1:12).EQ.'            ') THEN
                FAILPR(IPNT)=FAILPR(FAILPR(41)-1)
                J=(IPNT-1)*20+1
                K=(FAILPR(41)-1)*20+1-20 !last element on fail list
                FAIL(J:J+19)=FAIL(K:K+19)
                TRIED(IPNT)=TRIED(FAILPR(41))
                IPNT=FAILPR(41)-1
                FAILPR(41)=FAILPR(41)-1
                REASON(1:8)='        '
        ELSE
                FAILPR(IPNT)=ISAVE
```

```
              J=(IPNT-1)*20+1
              FAIL(J:J+11)=TSKOUT(1:12)
              FAIL(J+12:J+19)=REASON(1:8)
      ENDIF
      RETURN
      END
C=====================================================================
      SUBROUTINE TSKSWAP(IHR1,IHR2,SCH,SCHPR)
C=====================================================================
C Interchanges activites between two hour slots (1-40) in a a schedule
C
C
      CHARACTER SCH*480,TSKTEMP*12
      INTEGER SCHPR(40)
      J= (IHR1 - 1) * 12 + 1
      K= (IHR2 - 1) * 12 + 1
      TSKTEMP(1:12) = SCH (K:K+11)
      SCH(K:K+11) = SCH (J:J+11)
      SCH(J:J+11) = TSKTEMP(1:12)
      I=SCHPR(IHR1) !swap cdr priorities for respective tasks
      SCHPR(IHR1)=SCHPR(IHR2)
      SCHPR(IHR2)=I
      RETURN
      END
C=====================================================================
      SUBROUTINE UNSCH
C=====================================================================
C  schedules low priority tasks requiring no resources and having
C   no temp relationships
C
      CHARACTER TSKRS*1000,PRC*3000,IN*1
      INTEGER RSTIME(50,40),RSQTY(50)
      CHARACTER TSK*12,INFILE*12
C
C
      INCLUDE 'COMMON FOR/LIST'
C
      WRITE(5,1)
1     FORMAT(/' ***** Completing Schedules with Misc. Activites *****')
C
C  read the table of resources required by tasks
C  only need to read once
C
      IF(IRDFLAG.NE.1) THEN
      IRDFLAG=1
      OPEN (UNIT=3,FILE='TSKRSRC.DAT',STATUS='OLD')
      DO I=1, 50
         J=(I-1)*20+1  !compute pointer c1-12=task c13-20=resource
         READ(3,10)   TSKRS(J:J+19)
10       FORMAT(/A20)
      ENDDO
      CLOSE (UNIT=3)
C
C
C Read precedence table into memory
C
         OPEN (UNIT=3,FILE='PRCDTBL.DAT',STATUS='OLD')
         DO I=1,3000,30
           READ (3,11) PRC(I:I+29)
11         FORMAT(/A30)
         ENDDO
```

```
          CLOSE (UNIT=3)
        ENDIF
C
C  call the auxilliary routine for each company
C
        CALL UNSCHAUX(SCHA, SCHPRA, FAILA, FAILPRA, LOCKEDA,
     $       TSKRS, PRC)
C
        CALL UNSCHAUX(SCHB, SCHPRB, FAILB, FAILPRB, LOCKEDB,
     $       TSKRS, PRC)
C
        CALL UNSCHAUX(SCHC, SCHPRC, FAILC, FAILPRC, LOCKEDC,
     $       TSKRS, PRC)
C
        CALL UNSCHAUX(SCHD, SCHPRD, FAILD, FAILPRD, LOCKEDD,
     $       TSKRS, PRC)
C
        CALL UNSCHAUX(SCHH, SCHPRH, FAILH, FAILPRH, LOCKEDH,
     $       TSKRS, PRC)
C
        RETURN
        END
C=====================================================================
        SUBROUTINE UNSCHAUX (SCH, SCHPR, FAIL, FAILPR, LOCKED, TSKRS, PRC)
C=====================================================================
C
C puts unscheduled activities on the  company's schedule  2:MAY85 djg
C
        CHARACTER SCH*480, FAIL*800, TSK1*12, TSK2*12, REL*6,
     $      TSKRS*1000, PRC*3000, DUMMY1*8
        INTEGER SCHPR(40), FAILPR(41), LOCKED(40), DUMMY2(40)
C
C look at each hour of the schedule and find the blanks
C
        DO IHR=1,40
        ITSK=(IHR-1)*12+1   !pointer to the task in schedule
        IF(SCH(ITSK:ITSK+11).EQ.'            ') THEN  !found a blank
C find a activity on the fail list with low priority and
C no resource needs or temp relationships
                DO IFAIL=1,FAILPR(41)-1
                J=(IFAIL-1)*20+1
                TSK1(1:12)=FAIL(J:J+11)
                IF(FAILPR(IFAIL).EQ.99) THEN
                   DO IRSC =1,50  !check for resource needs
                        ITSKRS=(IRSC-1)*20+1
                        IF(TSK1(1:12).EQ.TSKRS(ITSKRS:ITSKRS+11)) GOTO 100
                        IF(TSKRS(ITSKRS:ITSKRS+2).EQ.'END') THEN
C  check the temporal relationship table
                        DO ITM=1,100
                                ITMPT=(ITM-1)*30+1
                                IF (PRC(ITMPT:ITMPT+2).EQ.'END') GOTO 85
                                IF(TSK1(1:12).EQ.PRC(ITMPT:ITMPT+11)) THEN
                                        GOTO 90 !failure
                                ENDIF
                        ENDDO
C
C ok. made it through the tests now going to put it on the sch
C
85                      CONTINUE
                        CALL TSKSUB(SCH, SCHPR, IHR, TSK1, TSK2, FAIL,
     $                          FAILPR, IFAIL, DUMMY1, DUMMY2)
```

```
90                              CONTINUE  !either in a temp relat or end
                                ENDIF
                           ENDDO
100                        CONTINUE  !either task needs resources or list at end
                        ENDIF
                        ENDDO
        ENDIF
200     CONTINUE !task put on schedule so try other hours
        ENDDO
        RETURN
        END
C=====================================================================
C   End of code
C=====================================================================
```

APPENDIX B

SAMPLE PROGRAM RUN WITH DATA TABLES

Initial training schedule set
(Asterisks denote company-level priority activities)

SCHEDULE FOR COMPANY A

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	ARCRFT REC 1	FIRST AID 4	*NERVE AGNT 4	INSTALL M18	WEAR M17 MSK
9:00	USE MAP 1	MAINT M16 1	FIRST AID 10	MNTN M17 MSK	*MISSION SUPP
10:00	NERVE AGNT 1	*WORK CALL 1	INSPECTION 4	FIRST AID 1	ID TRUCKS
11:00	ARCRFT REC 2	DECON SKIN 1	MAINT M16 2	USE MAP 2	MAINT .45 1
13:00	BATTLE DRILL	POLICE AREA	MISSION SUPP	PREP M72 1	*FIRST AID 5
14:00	WEAR PRT CLT	DECON SKIN 2	FIRST AID 9	*MAINT .45 2	ID HAND GREN
15:00	MAINT M80 2	*INSPECTION 1	INSPECTION 2	MAINT M80 1	ID ARMOR VEH
16:00	MAG AZIMUTH	FIRST AID 6	*ID WEAPONS	*PREP M72 2	AREA MAINT 4

SCHEDULE FOR COMPANY B

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	*MISSION SUPP	MAINT .45 2	FIRST AID 6	FIRST AID 3	POLICE AREA
9:00	*ARCRFT REC 1	MAINT M80 2	*INSPECTION 1	USE MAP 1	WORK CALL 4
10:00	FIRST AID 4	WORK CALL 2	*USE COMPASS	PREP M72 1	NERVE AGNT 2
11:00	INSPECTION 2	DET GRID CRD	AREA MAINT 4	MNTN M17 MSK	AREA MAINT 2
13:00	INSPECTION 3	MAINT M16 2	AREA MAINT 1	*FIRST AID 5	FIRST AID 10
14:00	*NERVE AGNT 3	BATTLE DRILL	*POLICE AREA	USE MAP 2	DECON SKIN 4
15:00	ID TERR FEAT	*ID ARMOR VEH	WORK CALL 3	ID HAND GREN	FIRST AID 2
16:00	INSTALL M18	FIRST AID 9	FIRST AID 7	DECON SKIN 2	NERVE AGNT 1

SCHEDULE FOR COMPANY C

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	*FIRST AID 3	AREA MAINT 3	MAINT M80 2	WEAR PRT CLT	INSPECTION 4
9:00	NERVE AGNT 3	*FIRST AID 9	FIRST AID 10	INSPECTION 3	*INSTALL M18
10:00	NERVE AGNT 2	FIRST AID 5	*NERVE AGNT 1	*PREP M72 1	ARCRFT REC 1
11:00	*WORK CALL 3	*DECON SKIN 3	MISSION SUPP	DECON SKIN 4	AREA MAINT 1
13:00	WORK CALL 2	MISSION SUPP	WORK CALL 4	USE MAP 1	FIRST AID 1
14:00	MAINT M80 1	INSPECTION 2	POLICE AREA	DET GRID CRD	ID HAND GREN
15:00	AREA MAINT 2	WEAR M17 MSK	ID WEAPONS	MNTN M17 MSK	ID TRUCKS
16:00	ID ARMOR VEH	FIRST AID 2	*USE COMPASS	AREA MAINT 4	RCOG CB HZRD

SCHEDULE FOR COMPANY D

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	*FIRST AID 2	*INSTALL M18	*ID WEAPONS	WORK CALL 1	FIRST AID 3
9:00	NERVE AGNT 2	MISSION SUPP	ARCRFT REC 1	*BATTLE DRILL	POLICE AREA
10:00	DECON SKIN 3	AREA MAINT 1	ARCRFT REC 2	INSPECTION 2	*MAINT M16 1
11:00	*FIRST AID 5	MAINT M80 2	AREA MAINT 2	NERVE AGNT 3	WORK CALL 2
13:00	FIRST AID 1	USE MAP 1	PREP M72 2	INSPECTION 4	MNTN M17 MSK
14:00	FIRST AID 4	DET GRID CRD	*ID HAND GREN	*FIRST AID 9	ID TERR FEAT
15:00	MAINT M80 1	ID ARMOR VEH	WORK CALL 3	POLICE AREA	USE MAP 2
16:00	FIRST AID 6	DECON SKIN 1	FIRST AID 7	WEAR PRT CLT	FIRST AID 10

SCHEDULE FOR HHC

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	NERVE AGNT 4	*NERVE AGNT 1	MAINT .45 1	MAG AZIMUTH	ID WEAPONS
9:00	FIRST AID 2	ARCRFT REC 1	POLICE AREA	ARCRFT REC 2	*ID TERR FEAT
10:00	*MISSION SUPP	INSPECTION 4	DECON SKIN 4	MISSION SUPP	MAINT M16 1
11:00	FIRST AID 5	FIRST AID 9	*ID ARMOR VEH	INSPECTION 1	*INSTALL M18
13:00	BATTLE DRILL	*MNTN M17 MSK	DECON SKIN 2	*INSPECTION 2	FIRST AID 8
14:00	ID TRUCKS	MAINT M80 1	*WORK CALL 1	FIRST AID 7	PREP M72 2
15:00	MAINT M80 2	WORK CALL 2	DECON SKIN 3	FIRST AID 1	FIRST AID 3
16:00	NERVE AGNT 2	WORK CALL 3	FIRST AID 4	RCOG CB HZRD	DECON SKIN 1

Training schedule set following inheritance of activities from higher echelons

SCHEDULE FOR COMPANY A

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	NERVE AGNT 4	INSTALL M18	PT
9:00	PERS HYGIENE	MAINT M16 1	FIRST AID 10	MNTN M17 MSK	MISSION SUPP
10:00	NERVE AGNT 1	M16 FIRING	INSPECTION 4	FIRST AID 1	COM TSK TEST
11:00	ARCRFT REC 2	DECON SKIN 1	MAINT M16 2	USE MAP 2	MAINT .45 1
13:00	BATTLE DRILL	POLICE AREA	MISSION SUPP	PREP M72 1	FTX 1
14:00	WEAR PRT CLT	DECON SKIN 2	FIRST AID 9	MAINT .45 2	FTX 2
15:00	MAINT M80 2	BAYONET PRAC	PARADE	RVW EDRE PLN	FTX 3
16:00	MAG AZIMUTH	FIRST AID 6	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR COMPANY B

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	FIRST AID 6	BAYONET PRAC	PT
9:00	PERS HYGIENE	MAINT M80 2	INSPECTION 1	USE MAP 1	WORK CALL 4
10:00	FIRST AID 4	WORK CALL 2	USE COMPASS	PREP M72 1	NERVE AGNT 2
11:00	INSPECTION 2	DET GRID CRD	AREA MAINT 4	RVW EDRE PLN	AREA MAINT 2
13:00	INSPECTION 3	MAINT M16 2	AREA MAINT 1	FIRST AID 5	FIRST AID 10
14:00	NERVE AGNT 3	BATTLE DRILL	POLICE AREA	USE MAP 2	DECON SKIN 4
15:00	ID TERR FEAT	ID ARMOR VEH	PARADE	ID HAND GREN	M16 FIRING
16:00	INSTALL M18	FIRST AID 9	PARADE	COM TSK TEST	NERVE AGNT 1

SCHEDULE FOR COMPANY C

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M80 2	WEAR PRT CLT	PT
9:00	PERS HYGIENE	FIRST AID 9	FIRST AID 10	RVW EDRE PLN	INSTALL M18
10:00	NERVE AGNT 2	FIRST AID 5	NERVE AGNT 1	PREP M72 1	ARCRFT REC 1
11:00	WORK CALL 3	COM TSK TEST	MISSION SUPP	M16 FIRING	AREA MAINT 1
13:00	WORK CALL 2	MISSION SUPP	BAYONET PRAC	USE MAP 1	FIRST AID 1
14:00	MAINT M80 1	INSPECTION 2	POLICE AREA	DET GRID CRD	ID HAND GREN
15:00	AREA MAINT 2	WEAR M17 MSK	PARADE	MNTN M17 MSK	ID TRUCKS
16:00	ID ARMOR VEH	FIRST AID 2	PARADE	AREA MAINT 4	RCOG CB HZRD

SCHEDULE FOR COMPANY D

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	ID WEAPONS	WORK CALL 1	PT
9:00	PERS HYGIENE	MISSION SUPP	ARCRFT REC 1	COM TSK TEST	POLICE AREA
10:00	DECON SKIN 3	AREA MAINT 1	ARCRFT REC 2	INSPECTION 2	MAINT M16 1
11:00	FIRST AID 5	MAINT M80 2	BAYONET PRAC	NERVE AGNT 3	WORK CALL 2
13:00	FIRST AID 1	USE MAP 1	PREP M72 2	INSPECTION 4	FTX 1
14:00	FIRST AID 4	DET GRID CRD	ID HAND GREN	FIRST AID 9	FTX 2
15:00	MAINT M80 1	RVW EDRE PLN	PARADE	POLICE AREA	FTX 3
16:00	FIRST AID 6	M16 FIRING	PARADE	WEAR PRT CLT	FTX 4

SCHEDULE FOR HHC

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT .45 1	MAG AZIMUTH	PT
9:00	PERS HYGIENE	M16 FIRING	POLICE AREA	ARCRFT REC 2	ID TERR FEAT
10:00	MISSION SUPP	INSPECTION 4	DECON SKIN 4	MISSION SUPP	MAINT M16 1
11:00	FIRST AID 5	COM TSK TEST	RVW EDRE PLN	INSPECTION 1	INSTALL M18
13:00	BATTLE DRILL	MNTN M17 MSK	DECON SKIN 2	INSPECTION 2	FIRST AID 8
14:00	ID TRUCKS	MAINT M80 1	WORK CALL 1	TEWT 1	PREP M72 2
15:00	MAINT M80 2	WORK CALL 2	PARADE	TEWT 2	FIRST AID 3
16:00	BAYONET PRAC	WORK CALL 3	PARADE	RCOG CB HZRD	DECON SKIN 1

Training schedule set following resource allocation

SCHEDULE FOR COMPANY A

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	NERVE AGNT 4	INSTALL M18	PT
9:00	PERS HYGIENE	COM TSK TEST	FIRST AID 10	MNTN M17 MSK	MISSION SUPP
10:00	FIRST AID 6	M16 FIRING	INSPECTION 4	FIRST AID 1	MAINT M16 1
11:00	ARCRFT REC 2	DECON SKIN 1	MAINT M16 2	USE MAP 2	MAINT .45 1
13:00	BATTLE DRILL	POLICE AREA	MISSION SUPP	PREP M72 1	FTX 1
14:00	WEAR PRT CLT	DECON SKIN 2	FIRST AID 9	MAINT .45 2	FTX 2
15:00	MAINT M80 2	BAYONET PRAC	PARADE	RVW EDRE PLN	FTX 3
16:00	MAG AZIMUTH	NERVE AGNT 1	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR COMPANY B

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	FIRST AID 6	BAYONET PRAC	PT
9:00	PERS HYGIENE	MAINT M80 2	INSPECTION 1	USE MAP 1	USE MAP 2
10:00	FIRST AID 4	WORK CALL 2	USE COMPASS	AREA MAINT 1	NERVE AGNT 2
11:00	INSPECTION 2	ID TERR FEAT	DET GRID CRD	RVW EDRE PLN	AREA MAINT 2
13:00	INSPECTION 3	MAINT M16 2	INSTALL M18	FIRST AID 5	FIRST AID 10
14:00	NERVE AGNT 3	BATTLE DRILL	POLICE AREA	ID ARMOR VEH	DECON SKIN 4
15:00	AREA MAINT 4	M16 FIRING	PARADE	ID HAND GREN	WORK CALL 4
16:00	PREP M72 1	FIRST AID 9	PARADE	COM TSK TEST	NERVE AGNT 1

SCHEDULE FOR COMPANY C

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M80 2	AREA MAINT 1	PT
9:00	PERS HYGIENE	FIRST AID 9	PREP M72 1	RVW EDRE PLN	AREA MAINT 4
10:00	NERVE AGNT 2	FIRST AID 10	NERVE AGNT 1	MAINT M80 1	ARCRFT REC 1
11:00	WORK CALL 3	COM TSK TEST	MISSION SUPP	MISSION SUPP	FIRST AID 1
13:00	WORK CALL 2	M16 FIRING	BAYONET PRAC	FIRST AID 2	POLICE AREA
14:00	FIRST AID 5	INSPECTION 2	ID ARMOR VEH	DET GRID CRD	ID HAND GREN
15:00	AREA MAINT 2	WEAR M17 MSK	PARADE	USE MAP 1	ID TRUCKS
16:00	WEAR PRT CLT	MNTN M17 MSK	PARADE	INSTALL M18	RCOG CB HZRD

SCHEDULE FOR COMPANY D

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M16 1	WORK CALL 1	PT
9:00	PERS HYGIENE	MISSION SUPP	AREA MAINT 1	COM TSK TEST	POLICE AREA
10:00	DECON SKIN 3	MAINT M80 2	ARCRFT REC 2	WEAR PRT CLT	INSPECTION 4
11:00	FIRST AID 5	ARCRFT REC 1	BAYONET PRAC	NERVE AGNT 3	WORK CALL 2
13:00	FIRST AID 1	USE MAP 1	ID HAND GREN	POLICE AREA	FTX 1
14:00	FIRST AID 4	DET GRID CRD	INSPECTION 2	FIRST AID 9	FTX 2
15:00	MAINT M80 1	RVW EDRE PLN	PARADE	ID WEAPONS	FTX 3
16:00	FIRST AID 6	M16 FIRING	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR HHC

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	INSTALL M18	WORK CALL 2	PT
9:00	PERS HYGIENE	M16 FIRING	FIRST AID 5	ARCRFT REC 2	MISSION SUPP
10:00	ID TERR FEAT	ID TRUCKS	DECON SKIN 4	DECON SKIN 2	MAINT M16 1
11:00	MAINT M80 2	MISSION SUPP	RVW EDRE PLN	DECON SKIN 1	MAINT .45 1
13:00	WORK CALL 3	BATTLE DRILL	COM TSK TEST	INSPECTION 2	FIRST AID 8
14:00	INSPECTION 1	MAINT M80 1	PREP M72 2	TEWT 1	POLICE AREA
15:00	FIRST AID 3	MAG AZIMUTH	PARADE	TEWT 2	WORK CALL 1
16:00	BAYONET PRAC	MNTN M17 MSK	PARADE	RCOG CB HZRD	

Training schedule set following rescheduling of company-level priorities

SCHEDULE FOR COMPANY A

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	NERVE AGNT 4	INSTALL M18	PT
9:00	PERS HYGIENE	COM TSK TEST	FIRST AID 10	MNTN M17 MSK	MISSION SUPP
10:00	FIRST AID 6	M16 FIRING	INSPECTION 4	FIRST AID 1	MAINT M16 1
11:00	ARCRFT REC 2	DECON SKIN 1	MAINT M16 2	WORK CALL 1	MAINT .45 1
13:00	BATTLE DRILL	INSPECTION 1	MISSION SUPP	PREP M72 1	FTX 1
14:00	WEAR PRT CLT	DECON SKIN 2	FIRST AID 9	MAINT .45 2	FTX 2
15:00	MAINT M80 2	BAYONET PRAC	PARADE	RVW EDRE PLN	FTX 3
16:00	MAG AZIMUTH	NERVE AGNT 1	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR COMPANY B

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	FIRST AID 6	BAYONET PRAC	PT
9:00	PERS HYGIENE	MAINT M80 2	INSPECTION 1	USE MAP 1	USE MAP 2
10:00	FIRST AID 4	WORK CALL 2	USE COMPASS	AREA MAINT 1	NERVE AGNT 2
11:00	INSPECTION 2	ID TERR FEAT	DET GRID CRD	RVW EDRE PLN	AREA MAINT 2
13:00	INSPECTION 3	MAINT M16 2	INSTALL M18	FIRST AID 5	FIRST AID 10
14:00	NERVE AGNT 3	BATTLE DRILL	POLICE AREA	ID ARMOR VEH	DECON SKIN 4
15:00	MISSION SUPP	M16 FIRING	PARADE	ID HAND GREN	WORK CALL 4
16:00	PREP M72 1	FIRST AID 9	PARADE	COM TSK TEST	ARCRFT REC 1

SCHEDULE FOR COMPANY C

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M80 2	AREA MAINT 1	PT
9:00	PERS HYGIENE	FIRST AID 9	PREP M72 1	RVW EDRE PLN	AREA MAINT 4
10:00	NERVE AGNT 2	FIRST AID 10	NERVE AGNT 1	MAINT M80 1	ARCRFT REC 1
11:00	WORK CALL 3	COM TSK TEST	MISSION SUPP	MISSION SUPP	FIRST AID 1
13:00	WORK CALL 2	M16 FIRING	BAYONET FRAC	FIRST AID 2	POLICE AREA
14:00	FIRST AID 5	DECON SKIN 3	ID ARMOR VEH	DET GRID CRD	ID HAND GREN
15:00	AREA MAINT 2	WEAR M17 MSK	PARADE	USE MAP 1	ID TRUCKS
16:00	WEAR PRT CLT	MNTN M17 MSK	PARADE	INSTALL M18	RCOG CB HZRD

SCHEDULE FOR COMPANY D

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M16 1	WORK CALL 1	PT
9:00	PERS HYGIENE	MISSION SUPP	AREA MAINT 1	COM TSK TEST	POLICE AREA
10:00	DECON SKIN 3	MAINT M80 2	ARCRFT REC 2	WEAR PRT CLT	INSPECTION 4
11:00	FIRST AID 5	ARCRFT REC 1	BAYONET PRAC	NERVE AGNT 3	WORK CALL 2
13:00	FIRST AID 1	USE MAP 1	ID HAND GREN	POLICE AREA	FTX 1
14:00	FIRST AID 4	DET GRID CRD	INSPECTION 2	FIRST AID 9	FTX 2
15:00	MAINT M80 1	RVW EDRE PLN	PARADE	ID WEAPONS	FTX 3
16:00	FIRST AID 6	M16 FIRING	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR HHC

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	INSTALL M18	WORK CALL 2	PT
9:00	PERS HYGIENE	M16 FIRING	FIRST AID 5	ARCRFT REC 2	MISSION SUPP
10:00	ID TERR FEAT	ID TRUCKS	DECON SKIN 4	DECON SKIN 2	MAINT M16 1
11:00	MAINT M80 2	MISSION SUPP	RVW EDRE PLN	DECON SKIN 1	MAINT .45 1
13:00	WORK CALL 3	BATTLE DRILL	COM TSK TEST	INSPECTION 2	FIRST AID 8
14:00	INSPECTION 1	MAINT M80 1	PREP M72 2	TEWT 1	POLICE AREA
15:00	FIRST AID 3	MAG AZIMUTH	PARADE	TEWT 2	WORK CALL 1
16:00	BAYONET PRAC	MNTN M17 MSK	PARADE	RCOG CB HZRD	NERVE AGNT 1

Training schedule set following resolution of temporal relationships

SCHEDULE FOR COMPANY A

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	NERVE AGNT 4	INSTALL M18	
9:00	PERS HYGIENE	COM TSK TEST	FIRST AID 10	MNTN M17 MSK	MISSION SUPP
10:00	FIRST AID 6	M16 FIRING	INSPECTION 4	FIRST AID 1	MAINT M16 1
11:00		DECON SKIN 1	MAINT M16 2	WORK CALL 1	TRP MOVEMENT
13:00	BATTLE DRILL	INSPECTION 1	MISSION SUPP	PREP M72 1	FTX 1
14:00	WEAR PRT CLT	MAINT M16 1	FIRST AID 9	MAINT .45 2	FTX 2
15:00	MAINT M80 2	BAYONET PRAC	PARADE	RVW EDRE PLN	FTX 3
16:00	MAG AZIMUTH	PERS HYGIENE	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR COMPANY B

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	FIRST AID 6	BAYONET PRAC	
9:00	PERS HYGIENE	MAINT M80 2	INSPECTION 1	PERS HYGIENE	USE MAP 2
10:00	FIRST AID 4	USE MAP 1	USE COMPASS	AREA MAINT 1	NERVE AGNT 2
11:00	NERVE AGNT 1	ID TERR FEAT	DET GRID CRD	RVW EDRE PLN	AREA MAINT 2
13:00	INSPECTION 3	MAINT M16 2	INSTALL M18	FIRST AID 5	FIRST AID 10
14:00	NERVE AGNT 3	BATTLE DRILL	POLICE AREA	ID ARMOR VEH	DECON SKIN 4
15:00	MISSION SUPP	M16 FIRING	PARADE	ID HAND GREN	WORK CALL 4
16:00	MAINT M16 1	FIRST AID 9	PARADE	COM TSK TEST	ARCRFT REC 1

SCHEDULE FOR COMPANY C

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M80 2	AREA MAINT 1	PT
9:00	PERS HYGIENE	FIRST AID 9	PREP M72 1	RVW EDRE PLN	PERS HYGIENE
10:00		FIRST AID 10	NERVE AGNT 1	MAINT M80 1	ARCRFT REC 1
11:00	WORK CALL 3	COM TSK TEST	MISSION SUPP	MISSION SUPP	FIRST AID 1
13:00	USE MAP 1	M16 FIRING		FIRST AID 2	POLICE AREA
14:00	FIRST AID 5	DECON SKIN 3	ID ARMOR VEH	DET GRID CRD	ID HAND GREN
15:00	MNTN M17 MSK	WEAR M17 MSK	PARADE	USE MAP 1	ID TRUCKS
16:00	FIRST AID 1	MNTN M17 MSK	PARADE	INSTALL M18	RCOG CB HZRD

SCHEDULE FOR COMPANY D

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M16 1	WORK CALL 1	PT
9:00	PERS HYGIENE	MISSION SUPP	AREA MAINT 1	COM TSK TEST	PERS HYGIENE
10:00	PREP M72 1	MAINT M80 2	ARCRFT REC 2	WEAR PRT CLT	INSPECTION 4
11:00	FIRST AID 5	ARCRFT REC 1		NERVE AGNT 3	TRP MOVEMENT
13:00	FIRST AID 1	USE MAP 1	ID HAND GREN	POLICE AREA	FTX 1
14:00	FIRST AID 4	DET GRID CRD	INSPECTION 2	FIRST AID 9	FTX 2
15:00	MAINT M80 1	RVW EDRE PLN	PARADE	ID WEAPONS	FTX 3
[illegible]:00	FIRST AID 6	M16 FIRING	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR HHC

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	INSTALL M18	WORK CALL 2	
9:00	PERS HYGIENE	M16 FIRING	FIRST AID 5	ARCRFT REC 2	MISSION SUPP
10:00	ID TERR FEAT	ID TRUCKS	DECON SKIN 1	DECON SKIN 2	MAINT M16 1
11:00	MAINT M80 2	MISSION SUPP	RVW EDRE PLN	DECON SKIN 1	MAINT .45 1
13:00	PREP M72 1	BATTLE DRILL	COM TSK TEST	INSPECTION 2	FIRST AID 8
14:00	INSPECTION 1	ARCRFT REC 1	PREP M72 2	TEWT 1	POLICE AREA
15:00	FIRST AID 3	MAG AZIMUTH	PARADE	TEWT 2	WORK CALL 1
16:00		MNTN M17 MSK	PARADE	RCOG CB HZRD	NERVE AGNT 1

Final training schedule set

SCHEDULE FOR COMPANY A

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	NERVE AGNT 4	INSTALL M18	MAINT .45 1
9:00	PERS HYGIENE	COM TSK TEST	FIRST AID 10	MNTN M17 MSK	MISSION SUPP
10:00	FIRST AID 6	M16 FIRING	INSPECTION 4	FIRST AID 1	MAINT M16 1
11:00	INSPECTION 2	DECON SKIN 1	MAINT M16 2	WORK CALL 1	TRP MOVEMENT
13:00	BATTLE DRILL	INSPECTION 1	MISSION SUPP	PREP M72 1	FTX 1
14:00	WEAR PRT CLT	MAINT M16 1	FIRST AID 9	MAINT .45 2	FTX 2
15:00	MAINT M80 2	BAYONET PRAC	PARADE	RVW EDRE PLN	FTX 3
16:00	MAG AZIMUTH	PERS HYGIENE	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR COMPANY B

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	FIRST AID 6	BAYONET PRAC	WORK CALL 3
9:00	PERS HYGIENE	MAINT M80 2	INSPECTION 1	PERS HYGIENE	USE MAP 2
10:00	FIRST AID 4	USE MAP 1	USE COMPASS	AREA MAINT 1	NERVE AGNT 2
11:00	NERVE AGNT 1	ID TERR FEAT	DET GRID CRD	RVW EDRE PLN	AREA MAINT 2
13:00	INSPECTION 3	MAINT M16 2	INSTALL M18	FIRST AID 5	FIRST AID 10
14:00	NERVE AGNT 3	BATTLE DRILL	POLICE AREA	ID ARMOR VEH	DECON SKIN 4
15:00	MISSION SUPP	M16 FIRING	PARADE	ID HAND GREN	WORK CALL 4
16:00	MAINT M16 1	FIRST AID 9	PARADE	COM TSK TEST	ARCRFT REC 1

SCHEDULE FOR COMPANY C

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M80 2	AREA MAINT 1	PT
9:00	PERS HYGIENE	FIRST AID 9	PREP M72 1	RVW EDRE PLN	PERS HYGIENE
10:00	NERVE AGNT 3	FIRST AID 10	NERVE AGNT 1	MAINT M80 1	ARCRFT REC 1
11:00	WORK CALL 3	COM TSK TEST	MISSION SUPP	MISSION SUPP	FIRST AID 1
13:00	USE MAP 1	M16 FIRING	AREA MAINT 4	FIRST AID 2	POLICE AREA
14:00	FIRST AID 5	DECON SKIN 3	ID ARMOR VEH	DET GRID CRD	ID HAND GREN
15:00	MNTN M17 MSK	WEAR M17 MSK	PARADE	USE MAP 1	ID TRUCKS
16:00	FIRST AID 1	MNTN M17 MSK	PARADE	INSTALL M18	RCOG CB HZRD

SCHEDULE FOR COMPANY D

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	MAINT M16 1	WORK CALL 1	PT
9:00	PERS HYGIENE	MISSION SUPP	AREA MAINT 1	COM TSK TEST	PERS HYGIENE
10:00	PREP M72 1	MAINT M80 2	ARCRFT REC 2	WEAR PRT CLT	INSPECTION 4
11:00	FIRST AID 5	ARCRFT REC 1	WORK CALL 3	NERVE AGNT 3	TRP MOVEMENT
13:00	FIRST AID 1	USE MAP 1	ID HAND GREN	POLICE AREA	FTX 1
14:00	FIRST AID 4	DET GRID CRD	INSPECTION 2	FIRST AID 9	FTX 2
15:00	MAINT M80 1	RVW EDRE PLN	PARADE	ID WEAPONS	FTX 3
16:00	FIRST AID 6	M16 FIRING	PARADE	PREP M72 2	FTX 4

SCHEDULE FOR HHC

TIME	Monday	Tuesday	Wednesday	Thursday	Friday
8:00	PT	BN INSPECTN	INSTALL M18	WORK CALL 2	NERVE AGNT 4
9:00	PERS HYGIENE	M16 FIRING	FIRST AID 5	ARCRFT REC 2	MISSION SUPP
10:00	ID TERR FEAT	ID TRUCKS	DECON SKIN 1	DECON SKIN 2	MAINT M16 1
11:00	MAINT M80 2	MISSION SUPP	RVW EDRE PLN	DECON SKIN 1	MAINT .45 1
13:00	PREP M72 1	BATTLE DRILL	COM TSK TEST	INSPECTION 2	FIRST AID 8
14:00	INSPECTION 1	ARCRFT REC 1	PREP M72 2	TEWT 1	POLICE AREA
15:00	FIRST AID 3	MAG AZIMUTH	PARADE	TEWT 2	WORK CALL 1
16:00	DECON SKIN 3	MNTN M17 MSK	PARADE	RCOG CB HZRD	NERVE AGNT 1

Data tables used for sample program run
(difficult problem)

```
 I----------I   List of possible
 1FIRST AID  1
 I----------I   acitivities.
 2FIRST AID  2
 I----------I   (TSKLST.DAT)
 3FIRST AID  3
 I----------I
 4FIRST AID  4
 I----------I
 5FIRST AID  5
 I----------I
 6FIRST AID  6
 I----------I
 7FIRST AID  7
 I----------I
 8FIRST AID  8
 I----------I
 9FIRST AID  9
 I----------I
10FIRST AID 10
 I----------I
11ARCRFT REC 1
 I----------I
12ARCRFT REC 2
 I----------I
13MISSION SUPP
 I----------I
14MISSION SUPP
 I----------I
15POLICE AREA
 I----------I
16POLICE AREA
 I----------I
17WORK CALL  1
 I----------I
18WORK CALL  2
 I----------I
19WORK CALL  3
 I----------I
20WORK CALL  4
```

```
 I----------I
21INSPECTION 1
 I----------I
22INSPECTION 2
 I----------I
23INSPECTION 3
 I----------I
24INSPECTION 4
 I----------I
25AREA MAINT 1
 I----------I
26AREA MAINT 2
 I----------I
27AREA MAINT 3
 I----------I
28AREA MAINT 4
 I----------I
29MNTN M17 MSK
 I----------I
30WEAR M17 MSK
 I----------I
31DECON SKIN 1
 I----------I
32WEAR PRT CLT
 I----------I
33RCOG CB HZRD
 I----------I
34MAINT M16 1
 I----------I
35MAINT M16 2
 I----------I
36MAINT M80 1
 I----------I
37MAINT M80 2
 I----------I
38MAINT .45 1
 I----------I
39MAINT .45 2
 I----------I
40PREP M72 1
```

```
  I----------I
41PREP M72 2
  I----------I
42ID HAND GREN
  I----------I
43ID ARMOR VEH
  I----------I
44ID TRUCKS
  I----------I
45ID WEAPONS
  I----------I
46INSTALL M18
  I----------I
47USE MAP 1
  I----------I
48USE MAP 2
  I----------I
49USE COMPASS
  I----------I
50ID TERR FEAT
  I----------I
51DET GRID CRD
  I----------I
52MAG AZIMUTH
  I----------I
53NERVE AGNT 1
  I----------I
54NERVE AGNT 2
  I----------I
55NERVE AGNT 3
  I----------I
56NERVE AGNT 4
  I----------I
57DECON SKIN 2
  I----------I
58DECON SKIN 3
  I----------I
59DECON SKIN 4
  I----------I
60BATTLE DRILL
```

```
I--task----II-resource   RESOURCE TABLE   max=50
RVW EDRE PLNINST DOE
I----------II------I      (TSKRSRC.DAT)
FIRST AID  2INST DOE
I----------II------I     If activity uses both
FIRST AID  3INST MOE
I----------II------I     expendable a renewable
FIRST AID  4INST MOE
I----------II------I     resources, expendables
FIRST AID  5INST ROE
I----------II------I     must be before renewables
FIRST AID  6INST ROE
I----------II------I     for code to work correctly!!!
FIRST AID  7INST JOE
I----------II------I
FIRST AID  8INST JOE
I----------II------I
FIRST AID  9INST COE
I----------II------I
FIRST AID 10INST COE  10
I----------II------I
M16 FIRING  M16 AMMO  11
I----------II------I
M16 FIRING  M16 RNG   12
I----------II------I
COM TSK TESTTR ARA 1  13
I----------II------I
ID TRUCKS   BN CLSRM  14
I----------II------I
ID WEAPONS  BN CLSRM  15
I----------II------I
NERVE AGNT 2BN CLSRM  16
I----------II------I
ARCRFT REC 2BN CLSRM  17
I----------II------I
BATTLE DRILLTR ARA 2  18
I----------II------I
USE COMPASS TR ARA 2  19
I----------II------I
INSTALL M18 TR ARA 3  20
```

```
I---task---I Unit      Short-range Calendar      (SRC.DAT)
PARADE       E 23
I----------I I Hour mandated.  0 is none
PARADE       E 24
I----------I I II    Unit E is every.    100 maximum will be read.
PT           E  1
I----------I I II   ***** FILE STRUCTURE NOTE *****
PERS HYGIENE E  2
I----------I I II    Activities with specified times MUST
PT           E 33
I----------I I II    preceed acitivies without specified times!!!
BN INSPECTN  E  9
I----------I I II
TEWT 1       H 30
I----------I I II
TEWT 2       H 31
I----------I I II
FTX 1        A 37
I----------I I II
FTX 2        A 38
I----------I I II
FTX 3        A 39
I----------I I II
FTX 4        A 40
I----------I I II
FTX 1        D 37
I----------I I II
FTX 2        D 38
I----------I I II
FTX 3        D 39
I----------I I II
FTX 4        D 40
I----------I I II
COM TSK TEST E  0
I----------I I II
RVW EDRE PLN E  0
I----------I I II
M16 FIRING   E  0
I----------I I II
BAYONET PRAC E  0
I----------I I II
END
```

```
I--task----II-resource
RVW EDRE PLNINST DOE
I----------II------I
FIRST AID  2INST DOE
I----------II------I
FIRST AID  3INST MOE
I----------II------I
FIRST AID  4INST MOE
I----------II------I
FIRST AID  5INST ROE
I----------II------I
FIRST AID  6INST ROE
I----------II------I
FIRST AID  7INST JOE
I----------II------I
FIRST AID  8INST JOE
I----------II------I
FIRST AID  9INST COE
I----------II------I
FIRST AID 10INST COE  10
I----------II------I
M16 FIRING  M16 AMMO  11
I----------II------I
M16 FIRING  M16 RNG   12
I----------II------I
COM TSK TESTTR ARA 1  13
I----------II------I
ID TRUCKS   BN CLSRM  14
I----------II------I
ID WEAPONS  BN CLSRM  15
I----------II------I
NERVE AGNT 2BN CLSRM  16
I----------II------I
ARCRFT REC 2BN CLSRM  17
I----------II------I
BATTLE DRILLTR ARA 2  18
I----------II------I
USE COMPASS TR ARA 2  19
I----------II------I
INSTALL M18 TR ARA 3  20
```

RESOURCE TABLE   max=50

(TSKRSRC.DAT)

If activity uses both expendable a renewable resources, expendables must be before renewables for code to work correctly!!!

```
I----------II------I
ID ARMOR VEHDEMO KIT  21
I----------II------I
FIRST AID  9BN CLSRM  22
I----------II------I
FIRST AID 10BN CLSRM  23
I----------II------I
ID TRUCKS   DEMO KIT  24
I----------II------I
ID WEAPONS  DEMO KIT  25
I----------II------I
ID TERR FEATTR ARA 2  26
I----------II------I
MAG AZIMUTH TR ARA 2  27
I----------II------I
ID HAND GRENDEMO KIT  28
I----------II------I
MAINT M80 2 TR ARA 2  29
I----------II------I
RCOG CB HZRDBN CLSRM  30
I----------II------I
END
```

```
I------I!!I--RESOURCE AVAILABILITY (RSRC.DAT)----I
INST DOE                  111111111111111111111111
resource!!I---+----!----+----!----+----!----+----I
BN CLSRM  1111111111111111111111111111111111111111
I------Iquantity if expendable, OR+----!----+----I
M16 AMMO05
I------I!!hours available if renewable-!----+----I
TR ARA 1  11111111111111111111111111111111
I------I!!I---+----maximum of 50 lines-!----+----I
M16 RNG           11111111
I------I!!I---+----!----+----!----+----!----+----I
INST MOE  11111111111111111111111111111111
I------I!!I---+----!----+----!----+----!----+----I
INST ROE  11111111        111111111111111111111111
I------I!!I---+----!----+----!----+----!----+----I
INST JOE  1111111111111111        1111111111111111
I------I!!I---+----!----+----!----+----!----+----I
INST COE          11111111111111111111111111111111
I------I!!I---+----!----+----!----+----!----+----I
TR ARA 2  11111111111111111111
I------I!!I---+----!----+----!----+----!----+----I
TR ARA 3          111111111111111111111111
I------I!!I---+----!----+----!----+----!----+----I
DEMO KIT                     111111111111111111111
I------I!!I---+----!----+----!----+----!----+----I
END
```

Table of precedence relationships: tsk (c12) relation (c6) tsk (c12)

(PRCDTBL. DAT)

```
PT          IMMBEFPERS HYGIENE
I----------II----II----------I
FTX 1       IMMAFTTRP MOVEMENT
I----------II----II----------I
ARCRFT REC 2AFTER ARCRFT REC 1
I----------II----II----------I
FTX 2       IMMFOLFTX 1
I----------II----II----------I
FTX 3       IMMFOLFTX 2
I----------II----II----------I
FTX 4       IMMFOLFTX 3
I----------II----II----------I
BAYONET PRACIMMBEFPERS HYGIENE
I----------II----II----------I
FIRST AID  2AFTER FIRST AID  1
I----------II----II----------I
WEAR M17 MSKAFTER MNTN M17 MSK
I----------II----II----------I
MAINT M16 2 AFTER MAINT M16 1
I----------II----II----------I
USE MAP 2   AFTER USE MAP 1
I----------II----II----------I
PREP M72 2  AFTER PREP M72 1
I----------II----II----------I
NERVE AGNT 2AFTER NERVE AGNT 1
I----------II----II----------I
DECON SKIN 2AFTER DECON SKIN 1
I----------II----II----------I
DET GRID CRDAFTER USE MAP 1
I----------II----II----------I
END
```